Dealing with Time Varying Motion Blur in Image Feature Matching

By

Andrew Gibb

Department of Electrical and Electronic Engineering
University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering.

May 2018

Word count: Fifty-one thousand, one hundred and sixty-six.

Abstract

Motion blur is present in many images and can be due to many causes: From shaky hand held photographs, the panning of 24 frames-per-second feature film cameras, a broadcast camera following a sprinter, or a camera on an autonomous robot. Judicious choice of camera parameters, illumination, and object speed can mitigate motion blur in some circumstances, but often it is unavoidable, or even desirable. For example, in the particular case of feature film and broadcast video, some amount of motion blur is desired, as it aids the creation of the illusion of a moving object, given a succession of still images, presented rapidly.

For video analysis however, motion blur remains an obstacle. Much of the work to date in visual analysis, and particularly in image matching, has not addressed motion blur. In the cases where both images are similarly blurred, this is not problematic, as these images appear similar, and can readily be identified as such. However when the motion blur differs between frames, many existing approaches fail or offer significantly reduced performance.

This thesis presents experiments that verifies the model of motion blur, which relates unblurred images to blurred ones, as a rectangular filter. It then proposes a modification to phase correlation, which is based on this rectangular filter model of motion blur. This is shown to perform as well as the best existing methods from the literature. Finally, modifications to SIFT descriptor matching are proposed and tested. One of the methods increases the accuracy of correct matching of SIFT features by up to 60%, for the case of matching a non-blurred image region to a blurred one.

Dedication and acknowledgements

For Mum and Dad.

I am grateful to the BBC for allowing me the freedom to undertake this study as part of my work, and for funding my tuition. I am particularly indebted to my BBC supervisor Prof. Graham Thomas without whose tireless support and enthusiasm this work would have been poorer in every way. I am also very grateful to Prof. Dave Bull from the University of Bristol, who was relentlessly helpful and positive, and with whom I shared many enlightening conversations.

Thanks are due to the co-ordinators of the European Union 7th Framework project "Fascinate", and their legal team, and also to Sam Chadwick, for arranging the agreement without which I would not have been allowed to do this work at all.

Dr. Johannes Steurer and colleagues at ARRI and Alastair Bruce at BBC R&D all deserve thanks for providing equipment and space for the experiments on motion blur. Without their help the selection of cameras would have been much less interesting. Thanks too to Dr. Steffen Gauglitz of the University of California, Santa Barbara, for his help with the data set he published.

Jen Hawkins was particularly helpful when it came to learning the ways of Bristol University. The people from Bristol University Merchant Venturer's Building for help, encouragement, or interesting distraction: Drs. Richard Vigars and Aaron Zhang for guidance at the start. Drs. Pui Anantrasirichai, Tilo Burghardt, Sion Hannuna and Paul Hill for their guidance at important junctures. Drs. Osian Haines, David Gibson, and everyone else for many stimulating conversations.

The people I work with every day at BBC R&D are, without exception, fantastic. The inspirational, supportive, and *interesting* working environment is a place I am very grateful to be in, every day, and it is made so by the people. There are far too many to name them all here, but a few notable characters deserve a mention: The newly minted Dr. Al Hinde, for sharing the pain. Dr. James Weaver, for his always clear, helpful, and caring advice. Kevin Price and Dr. Frank Melchior, for sharing another perspective entirely.

And finally, to my partner Rae, who has been an unwavering source of help, support and encouragement. Thanks so much for helping me out of the lowest points. Thanks for being by my side at the highs. And thanks for all the biscuits.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

Table of Contents

List of Tables

List of Figures

Figure **Page**

Chapter 1

Introduction

Motion blur occurs in many images, for many different reasons. In live and pre-recorded TV, and feature films, it is often intentional, or at least unavoidable. Industrial applications of computer vision often attempt to control the lighting, cameras, and object speed so that there is little or no motion blur. But as applications of computer vision become more widespread, and take place in uncontrolled environments (for example, self-driving cars), it becomes necessary to design algorithms which are robust to motion blur.

When motion blur changes by a small amount between a pair of images of the same object taken from similar locations, methods designed without considering motion blur can work rather well — the images will be very similar arrays of pixel values. But this is not always the case. Images containing strongly differing motion blur must often be matched. For example, a sensor on a car moving at different speeds along a particular stretch of road, a visual search algorithm trying to match a shaky photograph to other, non-blurred images, or a broadcast camera tracking system trying to match image regions between fast-moving frames and stationary ones.

This Thesis explores two questions derived from problems such as these:

1. Does motion blur behave as conventionally modelled?

2. Can these models improve visual tracking in situations of differing motion blur, in real time?

1.1 Motivation

Large differences in motion blur between images occur in unconstrained environments, or where the computer is not the primary "audience" for the video or images. Broadcast video and feature film recordings are intended for a human viewer first, and machine second. This

is in contrast to a lot of computer vision research, where the parameters for the camera can be set for the machine, and the subjective quality discarded. Applications in robotics and manufacturing monitoring are good examples of this.

A paper by Mikolajczyk and Schmid [4] won the 2014 CVPR Longuet-Higgins Prize, which recognises papers from ten years ago which have had "significant impact on computer vision research". This paper was part of a short series with [5, 6] which created a robust assessment framework for image feature detectors and descriptors. This framework incorporated test data with a variety of image distortions, but no motion blur. Invariance to affine change, illumination change, and scale change have all become part of the lexicon of computer vision research, but robustness to motion blur is much more rarely seen (let alone invariance to motion blur.)

Virtual graphics for broadcast video and for films have become a huge industry over the past two decades or so. Automated and semi-automated tools are used to find the position and orientation ("pose") of the camera for every frame so that virtual graphics can be added in such a way that they appear to be fixed to the real objects in the scene. A recent paper by Barber et al points out [7] the tools used in practice by the feature film industry for finding camera pose are often manual. In feature film post production, a predictable workflow is more important than one with more automation, if that automation produces larger variance in the time to complete a task.

Live broadcast television has different requirements than feature films (and post-produced TV.) Organisations such as the BBC have very high quality standards. As such, visual tracking for live broadcast must be extremely reliable. The BBC is also compelled to provide good value for money. This means they seek cheaper software based systems to produce equivalent results to camera mounts incorporating sensors, which are accurate, but expensive. Sensor based systems can also be impractical for broadcast — it is often the case that the cameras are installed and operated by a different company, and the the BBC only has access to the video.

1.1.1 The Piero sports graphics system

The Piero sports graphics system [8] is based on a set of software libraries developed by BBC Research and Development for adding virtual graphics to broadcast video coverage of live sports events. The day-to-day management of Piero, interface design, and integration into the broadcast chain are done by Red Bee Media in partnership with BBC R&D. The computer vision parts of Piero are designed to operate on pan-tilt-zoom broadcast cameras, which are common at live sports events. It is increasingly common for the users of Piero only to have access to the video feeds of a sports event - they cannot physically access the cameras. Because of this, the system is designed to work on video only. Inertial measurements of camera pose,

and encoded lens parameters, are assumed to never be available. Over the years Piero has been adapted by various authors to operate on more and more sports, and in increasingly challenging visual conditions.

The core functions of Piero are to calibrate frames of running video, live and in real time, and to do so in a reference frame which includes some real, physical geometry (such as the locations of the lines on a football pitch.) This enables the interactive portion of the software to create virtual graphics which appear to be fixed to the real world. A virtual camera is created with geometry matching the calibrations, and the graphics are rendered into that camera. The output from the virtual camera is composited with the video frame. As the camera moves, the graphics appear fixed to the world.

The tracking in this software is based on methods described by Lucas and Kanade [9] and Shi and Tomasi[10] which has become known as the "KLT" method. This approach, and derived methods, are described in more detail in Section 2.6.2.

In live broadcast it is critical that the graphics and the calibration be reliable and accurate. Even when the system is not generating graphics for immediate transmission, real time operation is important. When analysis packages are being collected together (including different angles of view on a particular event, expert analysis, as well as virtual graphics) the operator of the system may wish to navigate around the recorded timeline randomly. It is important that any section the operator chooses can be calibrated and rendered with virtual graphics immediately, so they can decide whether this is an appropriate view, and move on.

Many adaptations beyond the standard methods described in the literature have been made to ensure that the calibration works reliably in live broadcast situations. This means adaptations to varying lighting, animated advertising signs, unpredictable camera movement, and others. Each adaptation will be described in the following section.

The next section will consist of a detailed overview of the operation of Piero. The section after will describe the specific problem which motivates the research in this Thesis.

Operation

This section will first describe the online tracking process, as it operates on a frame of video where the previous frame is already calibrated. Next the various special additions will be described, and finally methods for initialization will be described. A more detailed description of this tracking system is given in [11].

Because the system is only ever used to calibrate cameras which do not translate, a simplified world model can be used to keep track of point correspondences between frames. Once the calibration for a frame has been established, any new features added in this frame are projected onto a cube which is centred on the camera and which has a fixed orientation with

respect to the world coordinate system. The 3D coordinates of the feature on the cube are treated as the world positions which correspond to the feature locations, for the purposes of estimating the camera pose.

On-line tracking proceeds as follows: When a new frame arrives, the velocity and pose from the previous frame are used to compute an estimated pose for this frame. An image pyramid is formed by repeatedly applying a fast running-average low pass filter, and downsampling by a factor of two in both image dimensions.

The 3D positions of the points on the cube are projected into the frame using the estimated pose. The KLT registration algorithm is used to refine the image coordinates of the projected points by comparing their appearance between this frame and the previous one. The KLT algorithm is applied in a coarse-to-fine manner using the image pyramid. Broadcast video tends to change in focal length in a slow, smooth manner, so image pyramids from adjacent frames can be assumed to represent the same scale. This process results in a set of point correspondences between the current and previous frames. Each pair of corresponding points also maps to a location on the cube surrounding the camera.

RANSAC [12] is now used to reject outliers. Pairs of features are selected at random. An estimated pose is found by assuming the geometry of one feature matches perfectly, then finding the rotation and focal length needed to cause the smallest error in the location of the other feature. This provides an estimated camera pose. These camera poses are compared and feature points which contributed to dissimilar camera poses are discarded.

An iterative optimization process now runs to determine the pose of the camera given the correspondences between each remaining feature point and its corresponding 3D "world" position. It is usual for iterative optimization procedures to have a threshold for per-iteration improvement, to halt the process. Piero also allows a hard limit on the number of iterations, so that the amount of time spent by the optimization is limited. Once the pose has been found, a normal calibration step is completed.

The process described above is prone to the accumulation of small errors, or "drift". When drawing virtual graphics on real-world objects, this has the effect of the graphics appearing to move away from the object they should be fixed to. Drift is corrected for in Piero using a set of features which are treated as a canonical reference set. The reference set is generated by storing image patches every time new features are detected in a region. In the coarse-to-fine matching process, the coarser levels of the image pyramid are matched by comparing the current frame with the previous one. After a number of coarser levels, the current frame is compared with the canonical feature instead. The levels at which this can occur can be configured by the user. By storing a permanent version of the appearance of a patch, and a 3D location, drift is minimised. As will be discussed later, this poses a problem when the amount of motion blur

in current and stored patches is significantly different.

There are a number of additional operations which don't happen every frame: As the camera view moves around the scene, parts of the image will no longer contain features, so new ones must be detected. The feature detection step operates on a sub-section of the frame (in x and y) per frame (in time). So, if the camera has panned, then the feature-less region at one side of the image will be populated over the course of a small number of frames (usually 4). Harris corners are detected in the sub-section. Optionally, the cornerness score of Harris [13] or the eigenvalue-based quality score of Shi and Tomasi [10] can be used to filter features. If more features are found than required, then areas of the sub-section which are crowded with features are culled, leaving a somewhat even distribution of feature points across the frame.

In an early version of the system, feature points could be detected on moving foreground objects, which would later need to be removed using RANSAC. A feature has been added to detect regions of the image where the motion cannot adequately be explained by the movement of the camera, which can be subsequently be masked out when feature detection takes place. This step takes place after the pose of the camera has been established for this frame. Low-pass-filtered versions of the current and preceeding frames are overlaid, one having been warped to match the view of the other. Areas where the intensity differences are greater than a threshold are treated as being part of a moving object, and excluding from the feature detection step.

Visual Feedback

Figure 1.1 shows the diagnostic from a test of Piero on a long jump attempt. The top shows the input field (interlaced video is still common in broadcast sports. If either the odd or even field alone is selected, it can be used as a proxy for a frame, as long as the aspect ratio is accounted for.) The bottom shows the same field with diagnostic markup. The green grid is a visualisation of the cube. Whether or not the grid appears fixed is a helpful proxy when trying to judge the stability or accuracy of the tracking by eye. The yellow crosses and green boxes indicate tracked feature points. A yellow cross is a feature point which has been located in this frame, and has been found to be an inlier in the RANSAC process. (The cross is red if it's a RANSAC outlier.) The green box around the yellow point indicates this feature point is one of the canonical reference features. In this sequence, the system has been automatically filling in reference features. The filled-in green area partially covering the athlete indicates the non-camera motion mask. Note that this motion mask doesn't just cover the athlete; they appear to have a "shadow". This is a result of examining the difference between two frames. Not only does the appearance of the athlete in this frame not match the background in the previous frame, but the appearance of the athlete in the last frame also does not match the appearance of the current frame.

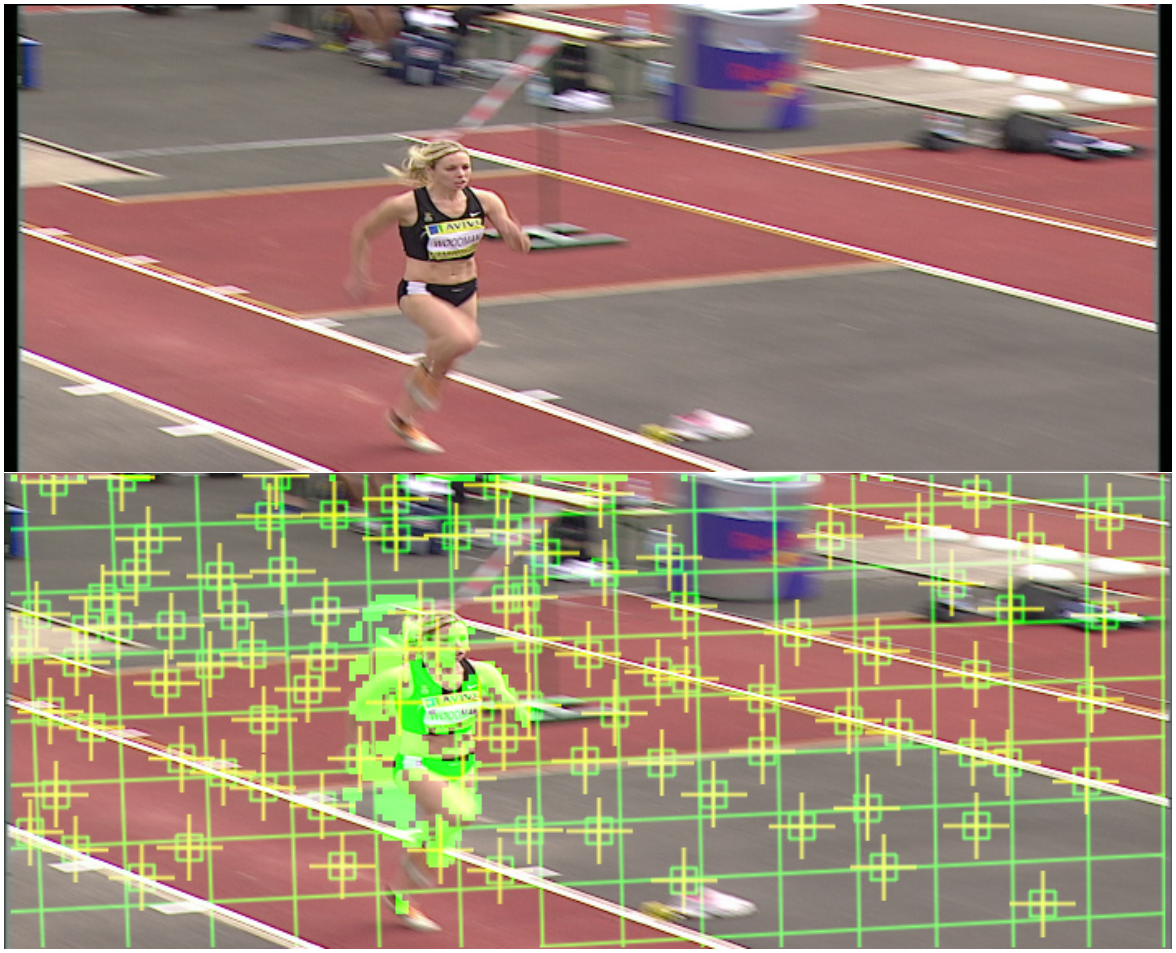


Figure 1.1: Top: Input field. Bottom: Piero output with diagnostic markup.

Practical use

Piero is used mostly for broadcast coverage of sports. Many sports, especially those which are broadcast on TV, have markings on the ground, which usually follow some more or less strict rules. Why not use these markings as calibration targets? Piero has this capability, although there are some practical limitations, which lead to the development of the image feature based method described above. Some structures are not as well-defiend as might be expected. For example, there is no standard dimension for an English Premier League football pitch. Also, football pictches are not flat. So in order to use the pitch markings as a calibration target, they must first be measured. Another problem when providing broadcast coverage for football is that the pitch lines are not always visible. A version of Piero capable or calibrating suitable shows using only pitch lines was developed earlier. It is described in detail it [14]. Quite often shots which need graphics cannot be calibrated using lines along because the shot only contained a single line segment. Finally, for athletics events, there are sometimes only parallel

lines. When viewing only a set of parallel lines, the position of the lines in the image will be almost totally invariant to a small camera movement along the lines. This is an example of the aperture problem, whereby the position along an edge viewed through a limited aperture is impossible to determine.

Lens distortion is ignored for broadcast sports applications. The way the system is used is robust to small errors of a few pixels, as long as the errors do not vary between frames.

Initialization

Two different approaches to initialization are available. If the Piero operator can control the view of the camera, then a manual initialization can be done. Otherwise an automatic calibration can be done. The automatic calibration is also used in case of emergency re-calibration.

Manual calibration involves arranging the camera view to include enough points with known 3D locations in the world (for example, measured corners on a football pitch. The operator indicates correspondences between image points and world points, and the camera pose is solved for.

Automatic calibration requires that the manual calibration has been carried out at least once, and some feature points have been stored. An exhaustive search is performed across a range of pan, tilt and zoom values, at specified intervals. For each set of pan, tilt and zoom parameters, the system attempts to match the visible features to the current frame using same approach as described above for a new frame. The best pose is taken as correct, and tracking is resumed. This procedure takes about a second, depending on the parameter values. The size of the intervals used depends upon the density of features stored, and the number of levels on the image pyramid.

Motivating examples

The following observation of the use of Piero for calibrating a camera showing a long jump attempt provided motivation for the work in this Thesis. During broadcast coverage of the long jump, the camera is positioned to the side of the sand pit, and some distance away. The camera pans to look at the athlete as they begin their approach run. As the athlete accelerates and moves closer the camera position, the rate of pan must increase to keep up. The camera is panning with maximum inmage plane velocity as the athlete makes their jump, then comes to an abrupt halt a very short time later as the athlete lands. The camera tracking is observed to oscillate once the camera stops panning. The overlaid graphics can be seen to move back and forwards by a few pixels. This is not satisfactory for broadcast use.

It is believed that this problem arises from the canonical reference features which are created as the sand-pit comes into view. Once the camera comes to rest on the sand pit, the system must match the (now sharp) features of the sand pit and surrounding area, using canonical features which are all blurred. (In this case, the user of Piero does not have control over the camera position. They only have a recording of the sequence to play, which they must attempt to calibrate, given just a few seconds to do so.) The work in this Thesis was begun with the initial aim of solving this problem.

1.1.2 The Fascinate Project

Fascinate [15, 16] was a collaborative project supported by the EU's 7th Framework. It ran from early 2010 until mid 2013 and involved 11 partner organisations from around Europe. The vision for the project was to create a "Format-agnostic, script-based production system" for live events. Such a system would capture a wide range of audio and video sources from a live event, annotate descriptive metadata alongside the audio and video, to create "scripts" which could render many different possible versions of the same live event dependent upon user preferences, device and network capabilities, and service provider control. This ambitious scope amounts to a complete repacement broadcasting system. Clearly it would be impossible to build such a system with a small project team in a few years. Instead, a vision of the complete system was used to inform a prototype design of the complete system, with each important subsystem represented. Over the course of the project, the consortium built and demonstrated this prototype.

Capture System

The vision of the capture system was to use many microphones and clusters of cameras [17]. A cluster of cameras would consist of several cameras with different capabilities, located as close together as possible. For example, there might be a wide-angle panoramic camera, a number of pan-tilt-zoom broadcast cameras, and a high frame-rate camera, all located near one another. In this example, the pan-tilt-zoom cameras would probably be manually operated, and would be following direction either from a human director, or some fixed guidelines. The panoramic camera would be stationary. Analysis algorithms or human operators would extract regions of interest and generate virtual shots by cropping the region from the panorama, with the various crops described only as metadata. The high frame-rate camera could be used as an input to analysis algorithms, or as part of the coverage if appropriate.

A feature of the project was the "Omnicam", developed by Fraunhofer HHI, which could capture a 180 degree panorama using 6 HD cameras and a mirror rig. After stitching, the

output resolution was 6000 × 2000 pixels. The Omnicam, installed in the Royal Albert Hall for a test shoot of the BBC Proms, is shown in Figure 1.2. An example frame showing the field of view of the Omicam is shown in Figure 1.3.

Figure 1.2: The Omnicam

It was specified in the project proposal document that dense 3D reconstruction would not be used, rather image-based rendering techniques based on calibration, segmentation, and warping should be used to combine information from the various cameras in the cluster. Since the intended used of the system was for coverage of live events, it was important to use real time processing with minimal latency everywhere.

Audio capture was carried out by the University of Salford and Technicolor. Various systems were used at different capture sessions including spatial audio capture with Soundfield Microphones and an "Eigenmic". Arrays of microphones with measured position captured audio information for localised parts of the scenes.

Figure 1.3: An example frame from the Omnicam

Automatic and semi automatic metadata generation

According to the vision of the system, the audio and video is processed to annotate metadata describing audio signals of interest, video regions of interest, common features, and so on. Some of this is done automatically, and some of it is operator assisted. All of the audio, video, and metadata, is then passed into the delivery system.

In the prototype a number of analysis systems were developed [18]. Systems were created by the BBC to locate players and the football in the various camera views during a football match using difference keying and some machine learning. Audio analysis was developed by the University of Salford and Technicolor to automatically extract audio objects such as the loudest part of the crowd at a football match, or the location of the kicked ball. Visual scene analysis was developed to allow detection of locally relevant semantic concepts (ie "people dancing", "person riding a bicycle") in a temporally stable way.

Data Network Considerations

The amount of data produced by the capture system is clearly too large for any existing large-scale delivery system to transmit, for a single event. Ten or twenty HD video streams, around forty audio signals, as well as all the attendant metadata, is far too much for any large-scale internet infrastructure to consider as "one channel" in the early twenty-first century. Doubtless that will change as time goes on, and for that reason the network capabilities to transmit such an enormous bandwidth were out of the scope of the project.

Some strategies for dealing with data in the network were investigated. For example, it seems plausible that the "last mile" of a network connection will remain low-bandwidth and heavily congested compared to the longer distance backhaul networks. Therefore, strategies for computing more or less of the rendering in the network were investigated. Tiled delivery of high-resolution video, remote control of an in-network renderer, and Publisher/Subscriber methods were all investigated by TNO and Alcatel-Lucent.

Script-based rendering system

The renderer's job is to receive all the media available (possible negotiating with a server to receive only relevant streams) along with the metadata and construct video and audio outputs determined by the user's preferences, service provider, and output device.

The final prototype renderer was able to render different parts of the Omnicam scene into a form suitable for display on a television or tablet. The crop could be chosen by the user, or could be driven by metadata describing a region of interest. This metadata might have originated automatically, or from a human operator. Suitable audio was rendered in a sophisticated way based on automatically extracted metadata, production-user input, and consumer input. The audio renderer was developed by the University of Salford. The video renderer, and renderer integration, was done by Technicolor. A video renderer was also developed by the author for some time during this project. Work was abandoned when it became clear the Technicolor renderer would be available to use for experiments.

Relevant problems

One motivation for this Thesis arises from the need for the Fascinate camera cluster to be calibrated. The panoramic camera is stationary, so stationary objects will be imaged without motion blur. Any object moving sufficiently fast will have motion blur. Note that changes to the camera aperture and exposure time are limited by the overall illumination of the scene, rather than to make crisp a single moving object. So, still things will be sharp, and moving things blurred.

Other cameras in the cluster will be able to pan, tilt and zoom in order to follow objects of interest. The exposure time and other camera parameters will be adjusted to achieve subjectively high quality pictures of the moving object. For an object moving sufficiently quickly, the background will be blurred, and the foreground will be sharp (mostly. Composite objects like humans might be blurred in parts owing to relative motion.) To calibrate these cameras into a common reference frame automatically, image features from both cameras must be matched. It would be sensible to use stationary background features - variation in illumination and appearance can be modeled over time to improve the stability of tracking over multi-day events. Therefore the calibration requires matching stationary background features from the panoramic camera (which are sharp) with those from the moving camera (which are blurred). And the calibration must be done in real time. It is possible to maintain camera calibration, once established, using only the video from that camera. But in order to prevent the calibration of the cluster of cameras from drifing apart, calibration of features between cameras must be done from time to time. There is no guarantee during a live event that any

particular camera will be stationary. Sometimes then it will be necessary to match blurred background features from the moving camera with sharp ones from the stationary camera. Hence, the problem investigated by this Thesis.

1.2 Contributions

There are three main contributions presented in this Thesis. The first is an experiment which, in this context, is used to verify the rectangular filter model of motion blur. Although this model is based on well-tested theoretical principles, no experiments have been carried out to verify that it is correct in this context. Experiments presented in this Thesis verify this model, and demonstrate the degree to which small degradations in the camera such as noise and small optical imperfections limit the precision of such measurements. The experimental design is found to be sensitive to image noise and spectral occupancy, two characteristics which map well onto subjectively good quality pictures. Therefore, the experimental method presented here can be used as a proxy for determining the overall quality of a camera and lens, in situations where image noise and spectral occupancy are critical.

The second contribution is a modification to Phase Correlation which permits accurate offsets to be found between signals degraded by motion blur, where ordinary phase correlation (and other methods - as discussed in Chapter 2) fail. This method is shown to have better performance than the state of the art method [19]. The method presented here offers the possibility of learning motion blur duration and can find larger offsets between images than [19]. This is not explored in detail, although preliminary findings are reported.

The final contribution is a collection of modifications to the procedure used to match SIFT features [20]. The best of these improves the number of correctly matches features between an image degraded by motion blur, and one unaffected, by up to 60%. The comparison of several different approaches sheds some light on the behaviour of SIFT features in the presence of motion blur. The work described in this Chapter has been submitted to ICIP 2017, but was not selected for inclusion. Publication is being sought elsewhere.

1.3 Outline of Thesis

This Thesis is structured as follows: A review of the literature is presented in Chapter 2. This covers previous attempts to deal with motion blur in both template-matching and detector-descriptor frameworks. The suitability of other published methods for modification to deal with motion blur is also discussed.

Chapter 3 describes the experiments to measure motion blur. Chapter 4 describes an experimental procedure to measure the performance of visual tracking methods in the presence of motion blur. Chapter 5 contains a description of the modification to Phase Correlation. Included are some observations based on the work carried out on this method, and an assessment of the performance of the method using the experimental procedure described in the previous Chapter. Chapter 6 contains a description of several possible modifications to SIFT feature matching to improve performance in the presence of motion blur. This method is also assessed using described experimental procedure. Finally, the Thesis concludes in Chapter 7. The contributions are assessed, and proposals for future work are given.

Chapter 2

Literature Review

2.1 Introduction

The main topic of this Thesis is the problem of matching image features from a sharp image to image features from an image with motion blur. Detecting locations in images which are both repeatable and stable with respect to image transformations (*image features*) has been addressed since at least 1980 [21]. Image features are useful in visual tracking, image and video search, classification, and data rate reduction as well as camera calibration. Describing these image feature points in a manner covariant with ([4]) any image transformations or distortions is crucial in any application where we wish to reproduce the human ability to identify an image of an object as the same object as in another image.

Computing correspondences between image features is important in the context of visual tracking, camera calibration, visual search, classification, and structure from motion. When the feature descriptor is a vector, it is conventional to choose between sum of absolute differences, Euclidean distance or Mahalanobis distance to determine how similar image features are. Occasionally specific applications will describe a novel matching method, or a particularly unusual descriptor will describe a custom matching procedure, as in [22] and [23]. The power of RANSAC [12] for finding a subset supporting a consensus hypothesis whilst rejecting non-supporters is so powerful that almost every method which involves feature matching uses it. In other cases, regions of images may be compared directly to determine if they match, and if they do, what the relative offset it. In this Thesism these latter methods are called "Template-based matching".

Despite this widespread base of computer vision applications, the majority of work reported ignores motion blur, although it is ubiquitous in amateur video and photography, broadcast and feature films, moving robots, autonomous cars, and the almost infinite variety of images and video on the internet. Most work on visual tracking employs an *instantaneous* camera

model, where objects do not move during the exposure. A few papers describe methods which accomplish visual tracking, or at least some kind of registration, in the presence of motion blur without explicitly modelling a moving camera. In general, these are less successful than those methods for tracking which attempt to take motion blur into account by using some kind of *integrating* camera model, where the relative motion of objects and the camera is accounted for.

2.1.1 Timing analysis

In attempting to find a new method for matching blurred and unblurred features in real time, the execution time of existing methods is crucial. Any modification to deal with motion blur is likely to involve additional work beyond the standard approach. A full tracking system such at Piero (Section 1.1.1) has many other jobs, all of which must also execute within the a frame period in order for real time performance to be achieved. Some offer a trade-off between their execution time and the time for the feature macthing process (eg, filtering of features to ensure they are sparsely distributed about the scene, and with a limit on the maximum number per frame)

When analysing whether a method might be suitable for a real time system, based on the description in a paper, a number of factors must be taken into account. Some authors provide timing information, and others do not. If timing information is not given, an estimate must be made. If timing information is given, then it is imperative to know what is accomplished in that time, and how is that relevant to incorporation in a tracking system. Additionally, some attempt should be made to assess how much effort has already gone into finding an optimal implementation. A GPU implementation is likely to be well adapted to the problem, as the APIs are low level. A Matlab implementation has the potential to be greatly improved upon as the interpreter introduces considerable overhead unless the code has been written with great care.

Each method described below will have a discussion on timing. As a rule of thumb, if a detection method can operate in around a frame period, then it's likely to be suitable for real time with some optimization. If a descriptor-matching method can perform around one hundred matches within a frame period, then it is likely to be suitable. A frame period is assumed to be $0.02s$, for a 25 Hz frame rate. Progressive broadcast video usually runs at 25 frames progressive or 50 fields interlaced. In either case, computing the camera parameters at 25Hz is suitable for live virtal graphics. A few methods in this review describe methods about an order of magnitude too slow, but were written ten or more years ago. These methods are generally considered suitable for use in a modern real time system, as CPU speeds have

gone up, and the problems for feature detectors and descriptor computation and matching are readily parallelizable, so they can can take advantage of modern multi-core CPUs.

This review is structured as follows: Some introductory sections describe the standard approaches to camera tracking, some widely used techniques, and key papers. Then, the remainder is divided into image feature detection, template-based tracking and descriptor-based tracking. The next section describes a paper with an integrated approach. Following that are section on deblurring and feature matching incorporating deblurring. The last Section summarises the findings.

2.2 Camera Tracking

Camera calibration is used for broadcast video live special effects, post-production virtual graphics, and augmented reality. This section describes the method used by the BBC R&D Piero system for live tracking of a pan-tilt-zoom camera, and also a model of camera tracking assumed for feature film post production. This model is based only on conversations with people working in the industry. Little is published by post production houses.

The approach used in Piero is to manually input a number of correspondences between 3D world coordinates and 2D image coordinates. From this information, the initial camera pose can be computed by triangulation. During this computation lens calibration information can be used to undistort the image and increase the accuracy of the tracking. (Not doen in Piero.)

Once initialized, the camera pose can be tracked from frame to frame by seeking corresponding points between frames. Figure 2.1 illustrates coarsely the processing in Piero. Drift can become a problem in camera calibration, as small errors accumulate. To counter this the software maintains a collection of "reference" features, which are treated as ground truth. Periodically, the camera pose is refined based on correspondences with the reference set. Broadcast camera motion tends to be smooth. As a result, motion blur tends to vary smoothly between frames, although it can still vary widely over a sequence. Drift correction which matches frames with strong motion blur against reference features without can cause significant errors in the tracking process. A number of optimisations have been added to this system to enable the tracking to be relatively stable for live broadcast use.

Camera calibration for feature film visual effects also uses feature correspondences and triangulation, but operates offline, so the computation has access to all frames in a sequence simultaneously. This means that global optimisations (eg [24]) can be applied. Frames which have been incorrectly calibrated are adjusted manually before virtual graphics are added. From conversations with staff at Double Negative, a post-production house, it is common for whole shots (of the order of 100 frames at 24 Hz) to require manual calibration, as a result of motion

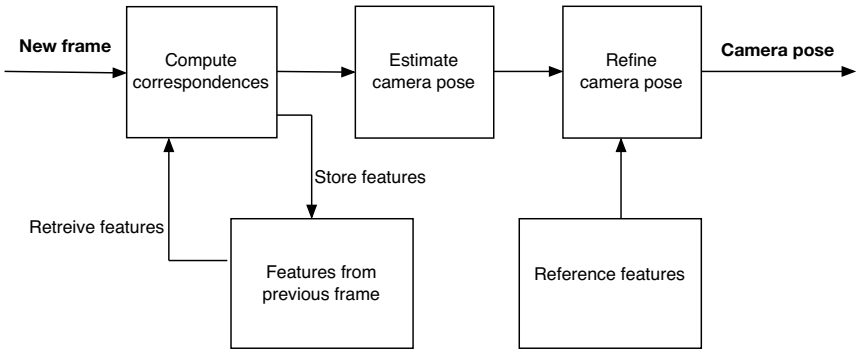


Figure 2.1: The main stages in the BBCs pan-tilt-zoom camera tracking software.

blur. Barber et al at Double Negative discuss [7] how significant numbers of man hours are spent computing the camera pose, and motion blur contributes significantly to the manual nature of this task.

2.3 Preliminaries

The concept of a Scale Space is used by a number of methods described in this review. RANSAC is used as a component in real-world matching systems. Both are important concepts and are introduced briefly below.

2.3.1 Scale Space

Some detectors and descriptors (eg [20, 9]) operate in "Scale Space" [25]. Typically, a scale space is found by applying a low pass filter to both dimensions of an image with a cut-off at $f_s/4$, then downsampling by a factor of 2 in both dimensions. This process is repeated for as many octaves as the application needs. The "Scale Space representation" of an image is the collection of all filtered and downsampled images, optionally including the original image. Sometimes, for example in SIFT [20], this general approach is modified. By localising a feature location in scale as well as image coordinates, some scale-invariance can be achieved, eg [6, 26].

2.3.2 Random Sample Consensus (RANSAC)

RANSAC [12] is a method for rejecting outliers in noisy data sets. The procedure is to pick a minimal random subset of the data, and determine the model parameters which those data imply. After a number of iterations, a particular set of model parameters will have a large number of supporting data. Those model parameters, along with the data which support them, are chosen and passed to the next stage of processing. The data which did not support the model can be discarded, or recorded as known bad data.

This method is applied to sets of correspondences in Piero tracking (See Figure 2.1) to eliminate spurious matches.

2.3.3 Camera Models

When dealing with motion blur, the relative motion between the scene and the camera must be taken into account. Methods not designed explicitly to deal with motion blur often use the "Instantaneous Camera" model, which assumes that the camera and all objects in the scene are not moving while the picture is made. Methods which explicitly attempt to deal with motion blur must assume some relative motion between the scene and the camera. In some cases, free motion of the camera and all objects in the scene can be modelled. In others, only scene-camera motion can be modelled. All elements of the scene are assumed relatively static, or even co-planar. All of these camera models are grouped under the term "Integrating camera".

2.4 Performance comparisons

A few key papers provide results which enable methods from different authors to be compared.

Mikolajczyk et al prepared [27] a comparison of region detectors which remains the key reference. They examine the impact of the following distortions:

- Angle of view (ie affine)
- Scale change
- In-plane rotation
- De-focus blur
- JPEG compression
- Illumination change

They assess detectors using metrics for repeatability and accuracy by comparing an undistorted image with one suffering a distortion. Repeatability is the proportion of features from a reference image which are re-detected in a distorted image. Accuracy is examined by measuring the proportion of features which match, using a standard feature-matching method. The assessment includes some careful consideration of sources of error. The experimental data and framework are published. Note that the impact of motion blur on the detectors is not considered.

Mikolajczyk et al published [4] a framework for measuring the performance of image feature descriptors under the same set of distortions as in [27]. Again, motion blur is not considered.

Descriptors were assessed in terms of recall vs 1-precision, defined as follows: Regions with similar descriptors are a *match*. Regions nearby in image space (after the distortion has been accounted for) *correspond*. If a feature point both *matches* and *corresponds* then it is a *true match*. If it *matches* but does not *correspond* then it is a *false match*.

Recall is the number of *true matches* divided by the number of *correspondences*. 1-precision is the number of *false matches* divided by the total number of *matches*. By varying the threshold below which two features *match*, curves are obtained showing how sensitive a particular descriptor is to the distortion in question.

Note that the number of correspondences is independent of the descriptor used - it is determined by the detector. Different detectors are used for different distortions with the aim of providing optimal input to the descriptor stage. More than one detector is used in some experiments to illustrate the suitability of the chosen detector.

Gauglitz et al produced [1] an analysis of detectors and descriptors, motivated by robotics. Their data covered a different set of distortions than [27]. They used multi-frame video sequences instead of image pairs:

- Angle of view (ie affine)
- Scale change
- In-plane Rotation
- Panning
- Motion blur
- Illumination change (both static and dynamic)
- Unconstrained

The Unconstrained sequences allow a more realistic tracking task to be assessed in the same framework as the isolated distortions. They re-use the performance metrics from Mikolajczyk [27, 4]. An additional performance metric "*reliability*" assesses the combined tracking ability of detector-descriptor pairs by comparison with a ground truth. To find reliability, the homography is computed between the frame and an abstract reference frame using the candidate correspondences. The average distance of four reference objects from ground truth is computed. If less than 5 pixels, the frame has been reliably tracked. Rates of reliably tracked frames per sequence are reported.

Some of Gauglitz et al's results on Motion Blur are included below. The results show performance of a descriptor or detector against motion speed. The camera begins at rest, and accelerates over a few frames up to full speed. The nine speed values are 1 to 9×5.1 pixels per frame.

2.5 Image Feature Detectors

Detecting stable image features, (sometimes "keypoints") is the starting point for any matching procedure. This Section describes a diverse set of techniques for detecting image features, and analyses their suitability for detecting features in the presence of motion blur.

2.5.1 Harris, Harris-Affine and Hessian-Affine

Harris corners [13] are widely used in vision systems with static cameras and lighting conditions. They have been generalized for use in more complex vision problems into the Harris-Affine and Hessian-Affine detectors. The basis of all methods is the so-called "second moment matrix", which is computed by first applying a Gaussian blur to the image, then finding the derivatives in x and y of the results. For the traditional Harris and Harris-Affine method, the second moment matrix $\mathbf{S}$ is given at each pixel $\mathbf{x}$ as

$$\mathbf{S}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 I(\mathbf{x})}{\partial x^2} & \frac{\partial I(\mathbf{x})}{\partial y}\frac{\partial I(\mathbf{x})}{\partial x} \\ \frac{\partial I(\mathbf{x})}{\partial y}\frac{\partial I(\mathbf{x})}{\partial x} & \frac{\partial^2 I(\mathbf{x})}{\partial y^2} \end{bmatrix}$$

Alternatively, the Hessian of the image at each pixel can be used in place of the second moment matrix:

$$\mathbf{S}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 I(\mathbf{x})}{\partial x^2} & \frac{\partial}{\partial y}\frac{\partial I(\mathbf{x})}{\partial x} \\ \frac{\partial}{\partial y}\frac{\partial I(\mathbf{x})}{\partial x} & \frac{\partial^2 I(\mathbf{x})}{\partial y^2} \end{bmatrix}$$

In the original Harris design [13], corners occur at local maxima of $R = \det(\mathbf{S}) - k\,\mathrm{tr}(\mathbf{S})$ above some threshold. k controls the tuning between corner and edge detection. In the Harris-Affine generalization introduced by Mikolajczyk and Schmid [28] maxima of R are sought in a

scale space created from the image. Then an iterative method is used to find warping (skew and stretch) parameters which define an affine-invariant representation of each point. As before the Hessian-Affine method substitutes the Hessian for the second moment matrix.

In their review paper, Mikolajczyk et al [27] found that the Harris-Affine and Hessian-Affine detectors were usually the strongest, with numbers of correct matches greater than any other detector in the majority of cases. None of the measures investigated caused the number of correct matches to fall to an unusably small number.

The computational cost of the Harris detector is $\mathcal{O}(n)$, for n pixels. The cost of the scale-selection and affine normalisation step is $\mathcal{O}((m+k)p)$ where p is the number of points found by the initial Harris detector, m is the number of scales in the scale-space, and k is the number of steps in the iteration. The Harris detector is considered suitable for real time feature detection.

No results are available that show how this family of feature detectors work under motion blur. Theoretically, the parameter k, and the threshold on R work together to prevent edges from being detected as features, because the feature position is ambiguous along the direction of the edge. Motion blur applied to the remaining corner-like features will increase the ambiguity in location of the feature, causing fewer features to be detected.

2.5.2 Difference of Gaussians

Lowe proposed [29, 20] a method to detect image features using the Difference of Gaussians. By incrementally convolving the image with Gaussians, a scale space is produced, which approximates the scale space of the scale-normalised Laplacian-of-Gaussians, [25]. Taking the difference between adjacent levels in the scale-space results in a set of images each convolved with a successively greater difference of Gaussians. Points which are local extrema in image coordinates and scale are the detected points.

In Gauglitz's [1] analysis the Difference of Gaussians detector performs relatively well, usually scoring better repeatability than the majority of the others tested. The only case when many other detectors outperform Difference of Gaussians is under perspective distortion. In Gauglitz's results, three of the other detectors outperform difference of Gaussians by more than 0.1 on their repeatability scale. For context, the best repeatability score for any detector under perspective distortion is 0.7. Under motion blur, the performance of this detector is second only to the Fast Hessian detector (Section 2.5.3).

Lowe's method requires multiple filtering operations on the scale space. Then the image differences must be computed. Bay et al [30] found that the run time for Difference of Gaussians was 400ms on an 640×480 pixel image. Since this was some years ago, it is likely that this detector is suitable for real time use today.

2.5.3 Fast Hessian

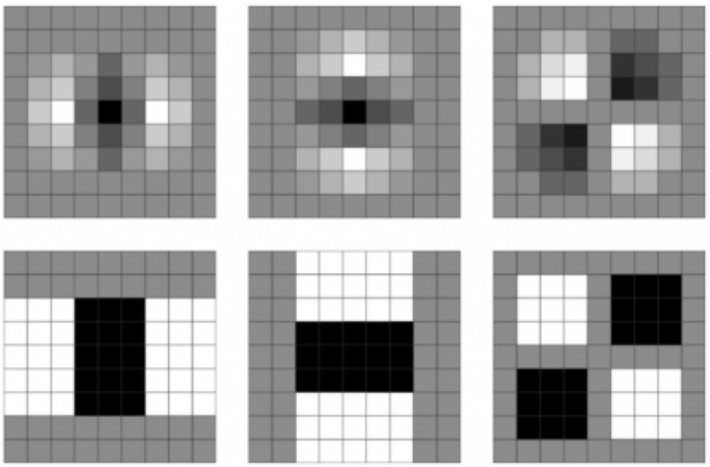


Figure 2.2: Three examples of approximations made by the Fast Hessian detector. Top row: Second order partial derivatives of Gaussians. Bottom row: Fast Hessian approximations.

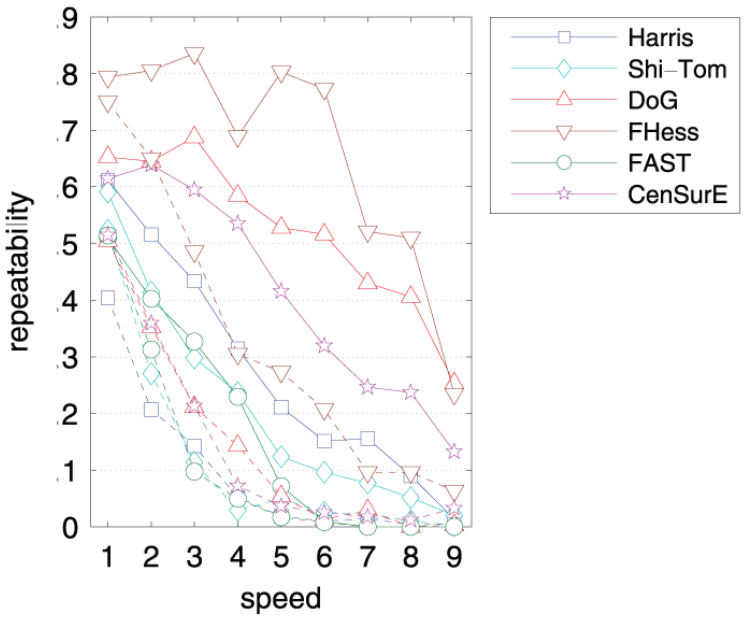


Figure 2.3: Gauglitz et al's [1] results for detectors under motion blur. Solid lines show matches between adjacent blurred frames. Dashed lines are matches between the first (unblurred) frame and later frames.

Bay et al introduced [30] the Fast Hessian detector, which utilises coarse approximations to earlier methods.

In a Hessian-based image feature detector, such as in Section 2.5.1, an optimisation can be made by exploiting the commutativity of the derivative operator. The Hessian of a Gaussian

is computed, then each image can be convolved with the Hessian of the Gaussian to give the Hessian of the image blurred with a Gaussian. Bay et al approximate the Gaussian with a much simpler filter, composed entirely of square blocks of pixels, each of which represents multiplication by an integer. Some examples are given in Figure 2.2.

The blocks of pixels are aligned with the sampling grid, so they can be computed using integral images. This allows for computation of a filter response in constant time, rather than time proportional to the number of pixels in the filter. In [30], Bay et al report that this detector runs in between 70 and 160 ms for a 640×480 pixel image. Since this was some years ago, it is likely suitable for real time use today.

In their own results, Bay et al [30] show the performance of the Fast Hessian detector to be better than every other detector in Mikolajczyk's analysis [27]. Subsequent analysis by Gauglitz et al [1] showed that the Fast Hessian detector was usually the most reliable of the detectors under test. (See Fig 2.3.) Of most interest to this work, Gauglitz found that the performance under motion blur was much better than any other detector.

2.5.4 Maximally Stable Extremal Regions (MSER)

The MSER detector was introduced by Matas et al [31]. MSER features are contours of constant intensity which contain pixels of either only greater or only lesser value, and which occur at a local minimum of change in region size. They are constructed by incremental growth from intensity extrema. Figure 2.4 shows some example MSERs.

Figure 2.4: Examples of Maximally Stable Extremal Regions, demonstrating affine invariance.

Matas et al reported in 2004 [31] that this detector would run in 0.14 s on a 530×350 pixel image. Since this was over ten years ago, it is likely this method will be useful for real time use today.

MSER performed well in the tests by Mikolajczyk et al [27]. Performance falls away sharply in the presence of defocus blur. This suggests that motion blur performance might also be poor, although no results supporting this have been published. Motion blur will change the distribution of intensity values around an image, potentially altering the sizes and shapes of MSERs.

2.5.5 Edge-based regions

This region detector was described in two papers co-authored by Tuytelaars and Van Gool [32, 33]. The method begins by finding Harris corners. Two strong edges are found at the corner, then an affine-invariant parameter is computed along the two edges. The corner and the points along either edge define a parallelogram. Two parameters are then examined, a function on the pixel values within the parallelogram, and another on the size and shape of the parallelogram. If an extremum is found in either of these functions, then an Edge-based feature point is recorded. Figure 2.5(a) indicates how and Edge-based region is formed.

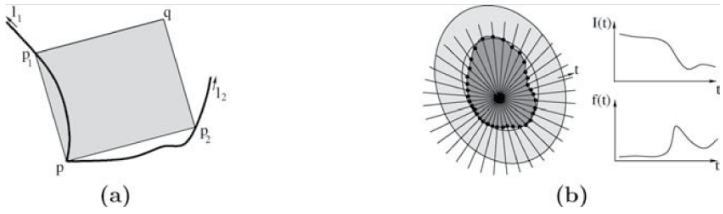


Figure 2.5: The grey quadrilateral in (a) is an Edge-based region. (b) Intensity-based region.

Mikolajczyk et al's results [27] showed the performance of the edge-based region to vary between the middle and the bottom of the distribution of results. Notably, for central blur of a textured scene, performance is similar across the whole range of values tested, although always returning fewer correct matches than other detectors tested. The number of correct matches falls off relatively slowly under JPEG compression. Under change of illumination, the results are similar for MSER, Edge-based regions, and both Hessian- and Harris-Affine.

The computational complexity of Edge-based regions is relatively high. Finding Harris corners is $\mathcal{O}(n)$, for n pixels. The second stage is $\mathcal{O}(pd)$, where p is the number of corners found in the first stage, and d is the average number of edges in the vicinity of a corner. Mikolajczyk et al reported [27] a run time approximately 100 times greater for Edge-based regions than for the Harris-Affine and Hessian-Affine methods. This is an offline method.

No results have been published demonstrating the performance of edge-based regions in the presence of motion blur. However, under sufficient motion blur an edge perpendicular to the direction of motion will become smoothed out such that a feature no longer appears. Further, a small section of a wiggly edge perpendicular to motion blur might change the values of the integral along the line. This detector is therefore not expected to perform well under motion blur.

2.5.6 Intensity-based regions

This method was described by Tuytelaars and Van Gool [32]. This method begins with an intensity extremum. Intensity values are shifted such that the value at the extremum is zero. A function is computed along radial lines which evaluates the intensity value divided by the integral of intensity up to that point. The region is defined by joining together extrema of this function. The authors note these extrema typically occur where the image gradient changes sharply. Figure 2.5(b) indicates how one of these regions is formed.

Mikolajczyk et al [27] mention this descriptor very briefly in their discussion. In most of their tests it has average performance. They observe that its performance varies with the material in the scene.

Computationally its performance is also average. It is slower to compute that Harris-Affine and Hessian-Affine, but not as slow as Edge-based regions and Salient regions (Section 2.5.7). Is is likely to be an offline method.

The effect of motion blur on these features will be two-fold. Locations of intensity extrema will become ambiguous, and the patterns of intensity extrema used to compute the function will be corrupted. This feature detector would not be expected to perform well under motion blur.

2.5.7 Salient regions

Kadir and Brady's Salient regions [34] search for extrema in entropy. First, the entropy is calculated at every pixel for a family of ellipses centred on that pixel, parameterized by radius, orientation, and ratio of major to minor size. The entropy scores for all ellipses are sorted. The n ellipses with the highest scores are salient features.

In their tests, Mikolajczyk et al [27] found Salient regions usually produced the fewest number of correct matches. Under scale changes, the number of correct matches falls very close to zero (it is hard to tell because of the size of the plot on the page.) Salient regions score lowest in the majority of tests.

In addition to relatively poor performance in detection, matching and tracking, Salient regions are very slow to compute, owing to the exhaustive computation of entropies required. Mikolajczyk et al reported an example computation time for one of their images, where Harris-Affine took 1.43 seconds, Salient regions took over 33 minutes. This is an offline method.

Given that all distortions applied by Mikolajczyk et al corresponded to a drop in performance of this feature, it seems unlikely that it will perform well in the presence of motion blur.

2.6 Template-based matching

Template-based methods seek matches between image regions using the pixel values directly. The methods in this Section work within this limitation to attempt to match images. Some attempt to modify the images by estimating model parameters for the distortions described in Section 2.2.

2.6.1 Phase Correlation

Phase correlation is an early method of determining an offset between two image regions. Applying this technique to find image alignment was proposed by Kuglin and Hines [35]. For a pair of image regions $f(\mathbf{x})$ and $g(\mathbf{x})$, phase correlation gives the offset between the images $\Delta\mathbf{x}$ as

$$F = \mathcal{F}(f(\mathbf{x})), \quad G = \mathcal{F}(g(\mathbf{x})),$$
$$\hat{S}_{fg}(\mathbf{u}) = \mathcal{F}^{-1}\frac{\bar{F}G}{|\bar{F}G|}$$

where $\mathcal{F}$ indicates the Fourier transform. For non-infinite signals, edge behaviour must be defined. Repeating support or constant values outside the main support are both used.

The timing analysis in Section 5.4 indicates that Phase Correlation can match several hundred pairs of image patches in a frame. Therefore, this method is suitable for real time use.

Phase correlation is widely used for image registration. Image differences in brightness and contrast are dealt with by normalisation, but changes in viewpoint, scale, and blur cause changes in the pixel values which will interfere with the matching. Applications including registration of aerial images, motion estimation for compression, and tracking the motion of a pan-tilt-zoom camera are suitable for phase correlation.

Squared-Power Phase Correlation

Ojansivu and Heikkilä observed [36] that by raising the cross power spectrum $\bar{F}G$ to an even power, the effect of any centrally symmetric filter upon phase is removed. The maximum offset which can be measured is equal to the size of the support divided by the power to which the cross power spectrum has been raised. This is not problematic in practice, as larger offsets cannot reliably be found using phase correlation.

The timing analysis in Section 5.4 also describes this method. Several hundred features can be compared in a frame. Therefore this is considered a real time method.

Their results [36] show that the RMS registration error remains small for regions of 300×300 pixels for motion blurs of up to 11 pixels in length. This is a promising method for registering small image regions subject to motion blur.

2.6.2 The "KLT" Method

The first widely used method for matching similar parts of images was the so-called Kanade-Lucas-Tomasi (KLT) tracker, which combined the methods described by Lucas and Kanade [9] with those from Shi and Tomasi [37]. Related work was collected together in a paper by Baker and Matthews [38], which focused on methods to compute the numerical part more efficiently.

Lucas and Kanade

Lucas and Kanade introduced [9] a gradient descent approach to matching similar regions between images. By assuming that the offset in image coordinates between images is small, the difference between the two images approximates the gradient of either image. A Newton-Raphson iterative gradient descent method can then find the relative offsets of the two images. When the images are separated only by offsets in image coordinates, using Newton-Raphson iteration is relatively straightforward. The method can also be applied to images separated by an affine homography. Estimating all parameters of this transform requires computing the Jacobian and Hessian of the parameter vector, which is expensive.

The method works well as long as the initial offset is less than a quarter wavelength of the highest frequency present in the signal. Offsets larger than a quarter wavelength can be found by using a scale-space, starting with a low-frequency estimate, and then refining by moving back up the scale space.

Neither Gauglitz et al [1] nor Mikolajczyk [27] give any results for this method. In internal tests, BBC R&D has found the KLT method to be effective for calibrating and tracking a pan-tilt-zoom broadcast camera, with some modifications [11]. It is well-suited to this problem because the static parts of the scene are approximately the interior of a sphere, with the

camera at the centre. Thomas et al published [11] results showing that the second derivative of pan remained small over a sequence of typical broadcast video, which means the tracking was smooth.

Good Features

Shi and Tomasi introduce [37] a method for selecting which image regions are worth tracking. The Lucas and Kanade method as described [9] encounters problems when image features were initialized on depth discontinuities or became occluded. Shi and Tomasi addressed both of these cases with a pair of criteria.

The first was to prevent low-contrast features from participating in the tracking. The eigenvalues of the covariance matrix at the feature should both be large. (Strictly, they must be within a few orders of magnitude of one another, and neither small.) This selects features which are high contrast and have broad frequency content in orthogonal image dimensions. It rejects low contrast features and those with highly directional information, like edges.

The second criterion is to reject features which become too dissimilar to the initial template against which they are tracked. This is done by warping each feature template to match the current version. If the difference is too great, or too variable, then the feature is removed from the analysis.

The camera tracker in the Piero sports graphics system uses this method with some enhancements [8, 39]. It has been found to work well with smoothly varying motion blur, and a camera which occasionally comes to rest. In internal testing, matching motion blurred frames with unblurred reference features can result in subjectively poor tracking results.

A Modern Framework

Baker and Matthews describe [38] a framework collecting together work building on [9]. They assume a generalized tracking problem where the input and template are related by an affine transform. The framework discriminates methods based on how the warping parameters are updated in the Newton-Raphson iteration. *Additive* methods compute an increment to the parameter vector $\delta\mathbf{p}$ which is added to the parameters from the previous iteration, $\mathbf{p}_1 = \mathbf{p}_0 + \delta\mathbf{p}$. *Compositional* methods compute an incremental warp which is composed with the previous warp, $W(\mathbf{p}_1) = W(\mathbf{p}) * W(\mathbf{p}_0)$. The framework also discriminates between forwards and inverse methods. Forwards methods iteratively improve a warp on the input until the difference with the template is small. Inverse methods warp the template to estimate an incremental warp, then compose the *inverse* of that warp with the input image, and iterate.

The Piero sports graphics system includes an implementation of the KLT method, incorporating the changes proposed by Shi and Tomasi [37], but not the model of Baker and Matthews. Without parallelisation, the Piero implementation was able to compare a small number of hundreds of features during a frame. This is then considered a real time method.

The key contribution of this paper is the inverse compositional method, which is much more efficient than the KLT (forward additive) method because it allows the Hessian of the update parameters to be precomputed. This efficiency gain increases as the square of the number of warping parameters. In the general affine case, this is significant, but for a pan-tilt-zoom camera, with only two warping parameters, the benefit is negligible.

2.6.3 Efficient Second-Order Minimisation (ESM)

Behnimane and Malis proposed [40] a different approach to avoiding the computation of the Hessian every iteration. They are able to replace the Hessian with the Jacobians of the parameters at $\mathbf{0}$, and the Jacobian of the current parameters by relying on the Lie Algebra representation of an affine transform which was introduced by Cipolla and Drummond [41]. As with the Inverse Compositional method (Section 2.6.2), this results in a greater increase in performance for an affine transform than a simpler one.

ESM with Pixel-wise Motion Estimation

Park, Lepetit and Woo describe [42, 43] an extension to ESM. They introduce an integrating camera into the parameter estimation step. They show that it is still possible to avoid computing the Hessian, and compute motion parameters along with the affine warp. In the latest iteration of their work [43] they also compute rolling shutter readout time duration.

The method works quickly. The authors give little detail on the implementation of the tracking part of their algorithm, but report that tracking takes 4.36 ms, which is fast enough for real time processing. The results presented show that the method is both reliable and accurate. In [42], the results indicate that for a sequence where ESM fails on 11 frames out of 752, this method fails on 5. Since ESM is essentially an optimisation on KLT, this performance is to be expected. Results showing behaviour when matching between an unblurred template and blurred input would have been more useful.

2.6.4 Simultaneous Minimisation of Motion Blur and Affine Parameters

Jin et al describe [44] a method which relies on the commutative nature of convolution. The problem they address is that of matching two image feature points which are related by an affine transform as well as differing motion blurs. Using the integrating camera model whereby

a motion blurred image I is some notional unblurred image I_s convolved with a filter $k_{\mathbf{v}}$, then for a template image I_0 and an input image I_t, both of which are corrupted with motion blur, and which are related by an affine transform, there should be a solution to the minimisation

$$\arg\min_{\mathbf{v}_0,\mathbf{v}_t,\mathbf{A}_t,\mathbf{d}_t} \sum_{\mathbf{x}\in\mathcal{W}} \|k_{\mathbf{v}_t} * I_0(\mathbf{x}) - k_{\mathbf{v}0} * I_t\left(\mathbf{A}_t\mathbf{x} + \mathbf{d}_t\right)\| \tag{2.1}$$

which simultaneously finds the image velocities $\mathbf{v}_{t,0}$, the affine transform $\mathbf{A}_t$ and the offset in image coordinates $\mathbf{d}_t$. Figure 2.6 shows an example of how this method affects a feature point over a video sequence. Observe that the image patches in the bottom row are a very close visual match for those in the top row.

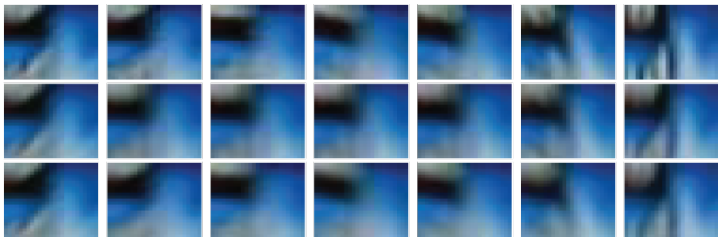


Figure 2.6: Top row: Image regions from a selection of frames in the video sequence. Middle row: The region from the first image motion blurred to match the top row frame. Bottom row: The region from the first image, motion blurred and warped to match the top row frame.

The results in [44] show a small number of sequences where the tracking is visualised. It performs well in the cases shown. RMS errors are given for seven matched regions, with KLT matching as a comparison [10]. Jin et al's results are much better, which is to be expected. Data showing tracking accuracy across a sequence is not provided, and would have been informative.

It is not clear whether this method would be suitable for real time tracking or not. The proposing paper [44] includes no timing information. The KLT method upon which this is based can compute a few hundred comparisons per frame, but this method includes many additional convolution operations to create the blurred image patches. This method is probably only suitable for offline operation. A very careful GPU-based implementation may be suitable for real time use, given the optimisations proposed by Mei and Reid [45].

Jin et al assert that the parameters for the affine transformation and those for motion blur should be estimated independently. Mei and Reid tested [45] this assumption by using the parameters of the affine transformation to calculate the direction of the blur, and estimating only one parameter for the length of the blur independently. Both approaches performed similarly in [45], except in the presence of noise, where the single parameter estimate resulting in more stable tracking.

2.6.5 Direction Detection and Exhaustive Search Blur Synthesis

Dai et al describe [46] a method which attempts to blur an unblurred template to match an input frame using an integrating camera model. For a new frame, a match with a template is sought using a mean-shift method. If this method fails, their blur-matching algorithm is invoked. Blur direction is determined using steerable filters. The unblurred template is then filtered with motion blurs of different lengths. A line search algorithm finds the best blur length. Mean-shift is used to match features with motion blur to this synthetically blurred template.

The results in the paper consist of six frames from video, with bounding boxes. The performance appears very good. It is too small a data set to draw any meaningful conclusions from. They give no indication of the computation time of their method. Given the number of steps required, it seems unlikely that a real time implementation will be possible.

2.7 Descriptor-based matching

Descriptors are representations of image regions which are not an array of pixel values. The goal is to create some representation which is the same for images of the same real object which suffer any or all of the distortions listed in Section 2.2.

2.7.1 Scale-Invariant Feature Transform (SIFT) and Related Methods

SIFT

Lowe's Scale Invariant Feature Transform [20] is inspired by how the early mammal visual system operates. The region around the image feature point is multiplied by a Gaussian, then divided into a square grid of subsections (typically 4×4). The grid is optionally aligned with the orientation of the image feature point. Each subregion is then further subdivided into 4×4 sample points. The image gradient is measured at each sample point, and a histogram (typically of eight bins) of image gradient direction in the subregion is computed. The feature descriptor vector is constructed by concatenating together the histograms from all 16 subregions. Figure 2.7 illustrates how the SIFT descriptor is constructed.

SIFT features perform very well in Mikolajczyk's analysis [4]. It is consistently near the top of the recall-precision curves.

In the analysis by Gauglitz et al [1] SIFT features perform relatively well under motion blur. In the single-image matching tests, the descriptor has the best or near-best matching scores under every distortion. In tracking tests, SIFT performed very well, with tracking

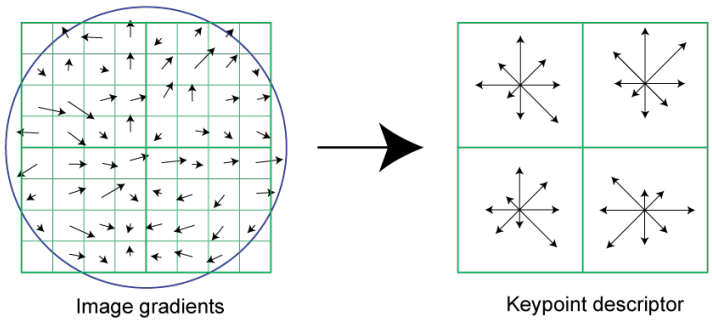


Figure 2.7: The accumulation of image gradients into histograms in the SIFT descriptor. The circle on the left indicates the presence of a Gaussian weighting over the region. For clarity, fewer regions are shown than used in practice.

performance which was more consistent across a longer motion blur than any other descriptor except Speeded-Up Robust Features [30] (Section 2.7.2).

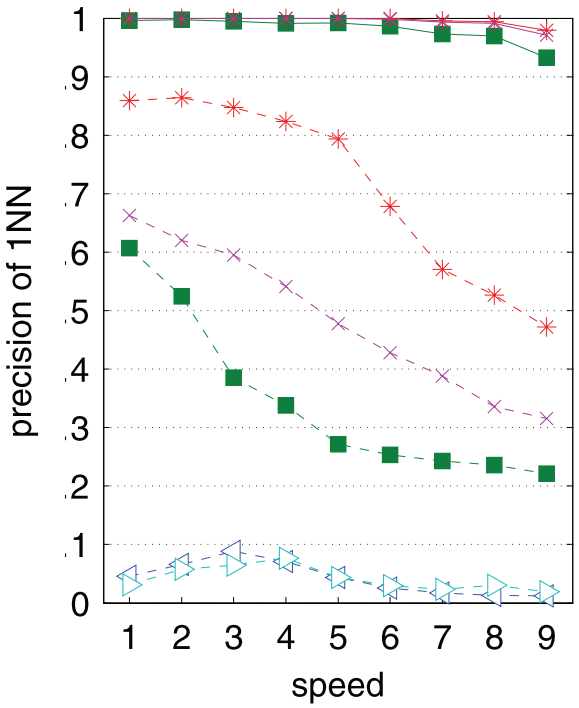


Figure 2.8: Gauglitz et al's [1] results for descriptors under motion blur. The star is SIFT, the cross SURF and the square phase correlation. The triangles are results for classifier-based methods. The dashed line indicates precision with respect to the first (unblurred) image in the sequence, the solid lines precision with respect to the previous frame in the sequence.

In his paper, Lowe observes [20] that "several thousand keypoints can be extracted from a typical image with near real time performance". A feature comparison requires 128 subtractions, and then the norm of the resulting 128-vector to be computed. This requires a trivial

amount of computation. This makes this method suitable for real time analysis.

If the feature descriptor is not oriented with the direction of the image feature point, but instead aligned with the sampling grid of the image, then there is a well-defined mapping between image direction and feature vector coefficient. This fact is exploited later in this Thesis in one method for dealing with motion blur when performing visual tracking.

Gradient Location and Orientation Histogram (GLOH)

Mikolajczyk and Schmid introduced [4] a new descriptor in their analysis paper. GLOH uses a similar approach to SIFT of computing gradient direction histograms of sub-regions around the image feature point. Instead of a square grid of sub-regions, GLOH divides the image region using a log-polar grid to compute histograms. They sample more, smaller regions, which results in a longer feature vector. This is reduced in size by principal component analysis (PCA) on a database of image features. The feature vector is constructed using the 128 largest eigenvectors. They do not describe the resultant covariance matrix.

Mikolajczyk and Schmid's results [4] for GLOH are similar to SIFT, and often a little better in terms of precision-recall. Timing performance is not given explicitly. However, this method is closely related to SIFT, which has already been classified as a real time method. The main differences are

- Computing a gradient histogram with a larger number of bins and
- Performing a matrix multiplication to convert from feature space to the space discovered in the PCA analysis.

Neither of these operations will vastly increase the computational cose beyond SIFT, and comparisons will cost almost the same as SIFT. Therefore, this is a real time method. Considering motion blur, the PCA step complicates the relationship between image directions and feature vector coefficients, compared to the more straightforward relationship observed in SIFT. Without knowing the covariance matrix, it is difficult to say what the effect of motion blur on this descriptor might be. Since motion blur is not included in [4], it might be assumed that there is no motion blur in the training data used to compute the covariance matrix.

PCA-SIFT

Mikolajczyk and Schmid [4] briefly discuss PCA-SIFT, due to Ke and Sukthankar [47]. This method samples many more subimages and constructs a much longer histogram than SIFT, then computes a PCA step similar to GLOH. The performance of PCA-SIFT in [4] is always close to average across all descriptors tested.

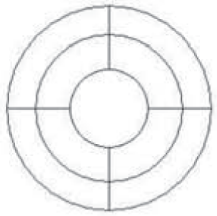


Figure 2.9: The arrangement of areas within which GLOH accumulates gradient histograms.

Computational complexity will be broadly similar to GLOH, so this is a real time method.

As with GLOH, the relationship between image directions and feature vector coefficient is only known if the covariance matrix is known. The ability of PCA-SIFT to operate in the presence of motion blur is, as with GLOH, difficult to say without knowledge of the covariance matrix.

Scale-Invariant Feature Detector with Error Resilience (SIFER)

Mainali et al [48] introduce SIFER, a modification to SIFT in which the filtering used to create the scale space is modified in order to distribute spacing more evenly between scale and space. They show results for their detector and descriptor working in this modified scale space for a number of image distortions. The results in [48] show better performance than SIFT or SURF when trying to register an image containing motion blur with one which does not.

The authors report timing performance of 3690 ms for this method on an image of 800×640 pixels. But in the same paper they report 1791 ms for SIFT and 1140 for SURF. This is much longer than suggested by the original authors of these two methods. Assuming that all implementations used for these experiments are equally optimised, this method takes about twice as long as SIFT. It seems likely this will be suitable for a real time system, although it might require more careful optimisation than others.

The authors note that the image features in the lower octaves are not found or matched reliably, and the matching process relies on features found in the higher octaves which are corrupted proportionally less by a given length of motion blur.

Iterative Affine and Illumination Matching

Yu et al introduced [49] a method for matching images from different viewpoints and illuminations. First, an affine transformation between a pair of images is estimated using correspondences derived using SIFT. That transformation is used to warp one image to match the other. Then histogram matching is performed to improve the illumination match between the

images. These steps are then iterated until the difference between the images is below some threshold, or a certain number of steps have been taken.

No results on motion blur are presented. This method seems unlikely to be useful in matching a motion blurred image to a sharp image, without modification to the SIFT matching process. Motion blur in one of the pair of images will also introduce errors into the histogram matching process. In the motion blurred image illumination levels of individual pixels will mix together, to produce a histogram with a different shape.

Since this method seems inappropriate for the main problem of this Thesis, no timing analysis was performed.

2.7.2 Speeded-up Robust Features (SURF)

Bay et al introduced [30] a feature descriptor which utilizes coarse approximations to filters, computed with integral images, to provide fast, approximate computation.

Given an image feature point location, the surrounding region is divided into a grid of 4×4 sub-regions. Within each sub-region, the Haar wavelet response is computed in x and y on a grid of 5×5 sample points. (The Haar wavelet is simply –1 for one half of the support, and 1 for the other half, divided in the centre. Clearly this is amenable to computation using integral images.) The responses in x and y are summed to provide the first two coefficients in the feature vector per sub-region. The magnitudes of the sum of the responses are the other two coefficients per sub-region. The feature vector is then constructed by concatenating together the wavelet responses and their magnitudes for each of the 16 sub regions, to give a feature vector 64 coefficients long. Figure 2.10 illustrates how the SURF descriptor is constructed.

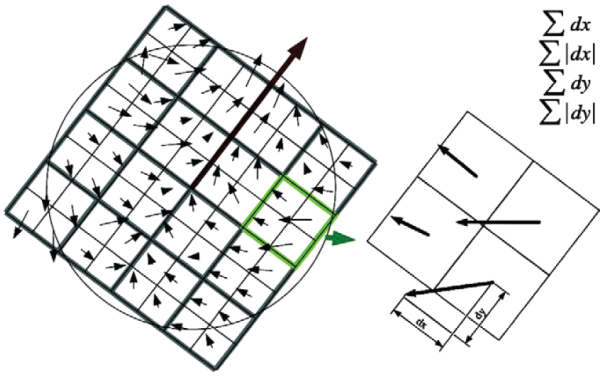


Figure 2.10: The construction of the SURF descriptor. Gradients accumulated in each sub-region are collected into four values.

In [30], Bay et al describe the time taken to carry out a combined detection, using the Fast Hessian detector, descriptor computation, and comparison on 1529 features in 610 ms, or 400 ms for upright SURF features. Since this paper is from 10 years ago, it seems likely that SURF

will be suitable for real time use today, particularly if used in conjunction with a strategy for limiting the number of features, as used in Piero.

Both the results in the original paper by Bay et al [30] and the analysis by Gauglitz et al [1] found SURF descriptors to have good performance. (See Figure 2.8.) Gauglitz et al also showed that SURF features perform relatively well in the presence of motion blur. They were relatively able to perform matches between adjacent motion blurred frames, and between the starting, non-blurred frame and a random selection of later, blurred frames.

2.7.3 Dual Tree Complex Wavelet Transform (DT-CWT) Multiscale Keypoints

Bendale et al proposed [50] a descriptor based on the DT-CWT [51]. In the ordinary wavelet transform the image is filtered by pairs of filters operating horizontally and vertically to produce a coefficient image which contains directional information about the image, localized in image co-ordinates, and a low-pass filtered image. The process is repeated iteratively on the low-pass filtered image to produce coefficients for lower octaves.

The wavelet transform is not shift-invariant: For a small change in the location of the signal with respect to the support, the wavelet coefficients will vary significantly and will not match under correlation. The DT-CWT introduces a second pair of filters (all four filters are designed together as a quad.) Analysis with these four filters produces a set of filter coefficients with properties analogous to the magnitude and phase of the Fourier transform. The magnitude of these coefficients is shift-invariant, and the phase encodes sub-wavelength shifts. The various combinations of high- and low-pass filters result in a family of six complex wavelets, each of which is directionally selective, and which together span all orientations. The results of DT-CWT analysis is therefore six complex subbands per octave.

Bendale et al adapt the descriptor introduced by Kingsbury [23] to multiple scales. Given an image feature point which is localized in image coordinates and scale, the descriptor is computed by concatenating together samples taken from all the subbands. Each subband is sampled twelve times around a circle of radius 1 centred on the image feature point as shown in Figure 2.11a. These are then assembled into matrix, as defined by the pattern shown in Figure 2.11b.

The result is a 2D array of DT-CWT coefficients in which image rotations corresponds to "cycling" the columns of the array.

No performance information is given by the original authors. Some informal experiments with the matlab code supplied by Prof Kingsbury (by direct request only) suggest that the execution time is too long for real time use. Therefore, this method is classified as an offline method.

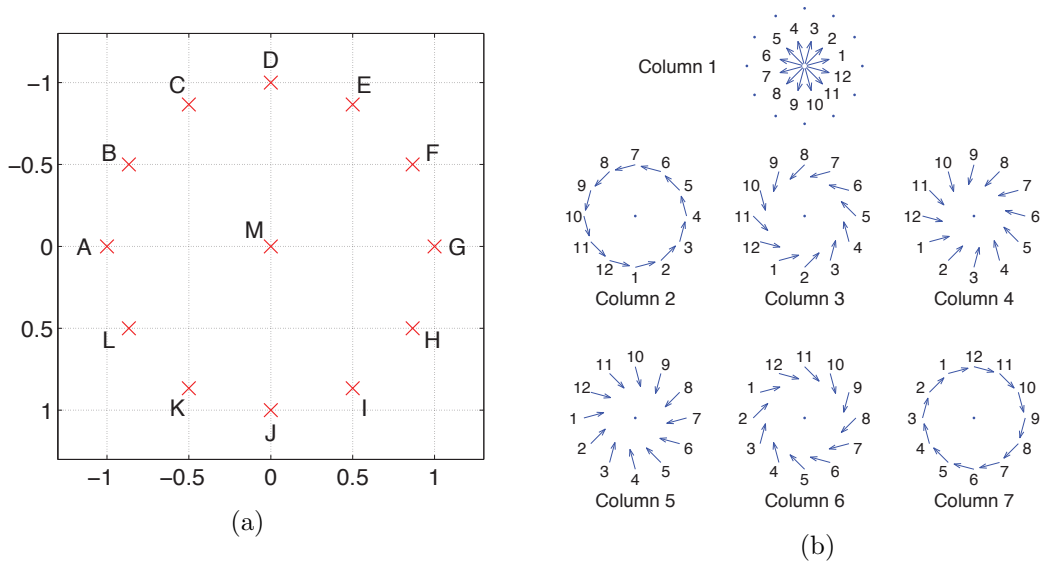


Figure 2.11: (a) The sampling structure used to extract DT-CWT coefficients in the vicinity of a feature point. (b) The arrangement of the extracted coefficients into a feature matrix.

No results were presented demonstrating the performance in the presence of motion blur. The effect of motion blur to corrupt phase as well as magnitude may well introduce significant errors in the subband coefficients. As with SIFT, there is a well-defined relationship between image direction and coefficient location means that the impact of motion blur can be easily understood and isolated, and potentially corrected for.

2.7.4 Phase-based Local Features

Carneiro and Jepson introduced [52] a phase-based image descriptor. The method uses a quadrature pair of filters. One filter is the second derivative of a Gaussian, and the other is the approximation of the Hilbert transform of the first. It seems that these filter kernels are specified in two dimensions, but either the Gaussian or the derivative (or both) is a function of one dimension, whose orientation varies to give a directional response. By using the Hilbert transform, the response to the filter pair can be interpreted as a complex value, $f + ih$, where f is the response to the Gaussian part, and h the response to the Hilbert transform part.

To compute the descriptor, first a sampling structure is defined around an oriented feature point, as illustrated in Figure 2.12. The feature point forms the centre of the circle, and the first sampling point. The second is a point r pixels away in the direction of the orientation of the feature point. n other points are evenly spaced in a circle of radius r pixels around the feature point. A number of filter pairs are applied at each feature point, with varying angles and standard deviations. The complex results of these are concatenated together into a feature

vector.

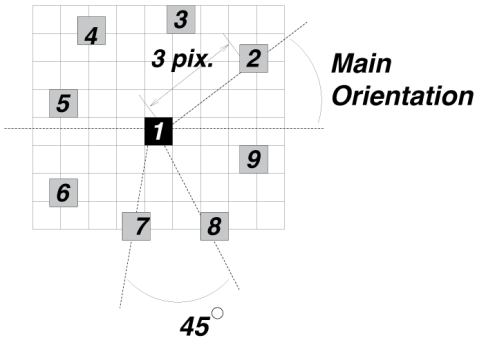


Figure 2.12: The selection of sample points in Phase-based local features.

The results in [52] show good performance in the presence of brightness changes, and worse performance in the presence of scale changes. There are no results describing performance in the presence of motion blur.

This method only requires the computation of n filter responses per image which, depending on n, might be fast enough for real time performance. The matching process is a comparison between two short vectors, so requires a trivial amount of computation. This is classified as a real time method.

The authors specify that the difference between two feature descriptors be computed as the normalised sum of the difference in phase at each feature coefficient. If these descriptors were computed oriented to the image sampling structure, then it would be possible to map from image directions to feature coefficient. This means that it may be possible to isolate the feature coefficients affected by motion blur.

2.7.5 Phase Quantization

Two papers have been found which attempt to provide immunity to motion blur by quantising phase. The first quantized phase of Fourier coefficients, and the second the phase of DT-CWT subband coefficients.

Small-Time Fourier Phase Quantization

Rahtu et al suggest [53] a method for texture classification based on the small-time Fourier transform of image regions. They examine image regions of between 3×3 and 11×11 pixels. The descriptor is computed as follows: A set of directional filters are applied to the image. Each filter produces a complex response at each pixel in the region. The phases of these complex responses are quantized to a 2-bit value. These values are concatenated, producing a codeword at each pixel. A histogram of all the codewords in the patch is built, and normalised.

They present results using other descriptors, under two different clustering methods. They only show detailed results for motion blur length of up to four pixels, after which their best performing descriptor has 75% classification accuracy, down from around 95%. The performance of their descriptors appears to fall off sharply when the motion blur length reaches 2.5 pixels. This might be a result of the motion blur phase inversion effect: Phases quantized into 4 bins after being inverted will still not match.

This method only requires a small number of filtering operations on the image per frame period. It is likely that a real time implementation would be possible.

The results presented in [53] only show motion blur up to four pixels long. Performance drops off significantly with increasing blur. The reader is left to speculate on whether increasing blur further will cause a proportional drop off in matching performance.

Undecimated DT-CWT Phase Quantisation

Anantrasirichai et al describe [2] a method for texture classification based on the Undecimated Dual-Tree Complex Wavelet Transform (UDT-CWT). They take the UDT-CWT of a candidate texture patch, and quantize the phase in each subband to create a 2-bit descriptor for each pixel. The descriptors are concatenated over octaves to give a codeword for each pixel. These codewords are then collected into a histogram to represent the image region.

Results are reported for descriptors computed up to the third and fourth octaves, and for motion blur of up to five pixels in length. The first octave is discarded, as it was not found to be useful. With no motion blur, classification accuracy of textures close to 100% is reported for both the authors method and others. At five pixels motion blur, the method using octaves two, three and four attains 90% accuracy, and the method using only levels two and three attains 65% accuracy.

Figure 2.13 shows the results from [2] for texture classification in the presence of motion blur. This plot shows poorer performance when using only octaves two and three, compared to two, three and four. The phase inversion effect of motion blur may provide an explanation.

Figure 2.14 shows how the phase inversion effect of motion blur maps onto the octaves of wavelet decomposition. The phase inversion effect of the five-pixel motion blur described in the paper only impacts the upper third or so of the energy in the second octave. Some proportion of the sample phases in this octave are likely to be inverted by motion blur. Exactly how many will depend on the direction of the motion blur, the distribution of gradient directions, and the spectral content at each sample location. Under the assumption that *some* proportion of the signal is corrupted, it is conceivable that the distance between descriptors in feature space is increased by this effect.

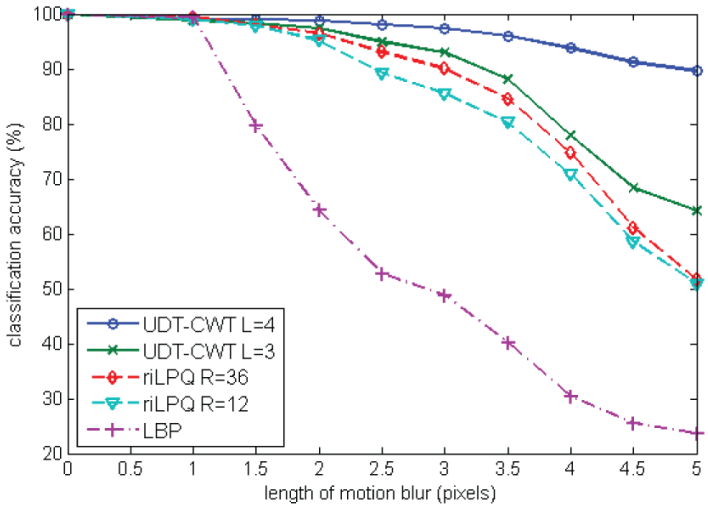


Figure 2.13: Anantrasirichai et al's results [2]. UDT-CWT L=3 uses subbands 2 and 3, L=4 uses subbands 2, 3 and 4.

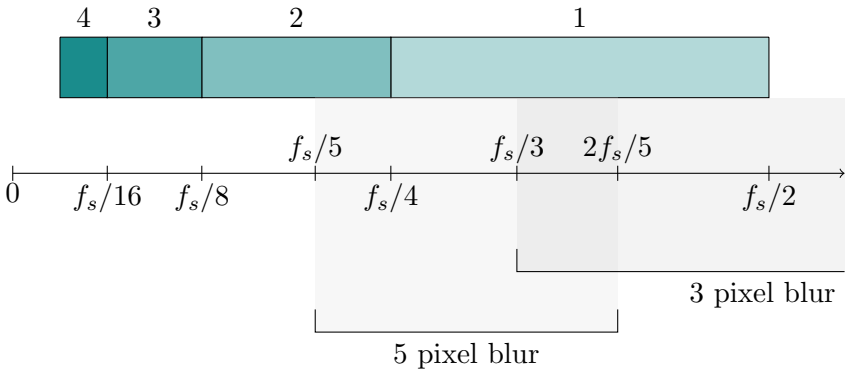


Figure 2.14: The relationship between short motion blur phase inversion and the first four octaves of a signal. The numbered regions at the top are the frequency bands extracted at the indicated octave by the (U)DT-CWT. The shaded regions at the bottom show which phases are inverted for 3- and 5-pixel motion blurs.

This method relies on the DT-CWT, which as noted above has yet to have a fast implementation. It is classified as an offline method.

The descriptor which includes the fourth octave includes 50% more data points than lower frequency descriptor, and these data are free from motion blur influence. The effect of the same number of corrupted sample phases will be considerably smaller on the overall distance between descriptors when the fourth octave is included.

2.8 Scale-Space Approximation

Pretto et al describe [54] a method to compute visual odometry for a legged robot using an integrating camera model. They compute the direction of motion blur by finding the minimum of the response to a set of directional high-pass filters. The extent is computed by finding the minimum of the auto-correlation along the blur direction. Blur parameters are computed for each pixel and clustered into rectangular subregions.

If motion blur is modelled as a convolution of some notional unblurred image $I(\mathbf{x})$ with a rectangular filter $k_{\mathbf{v}}(\mathbf{x})$, then it should be possible to find a filter $g'(\mathbf{x})$ which, when applied to the motion blurred image, approximates the effect of a Gaussian blur on the notional unblurred image:

$$g(\mathbf{x}, \sigma) * I(\mathbf{x}) \approx g'(\mathbf{x}, \sigma) * k_{\mathbf{v}}(\mathbf{x}) * I(\mathbf{x})$$

They use a numerical iterative method to approximate the filters $g'(\mathbf{x})$. g' is then used to compute an approximate scale-space. Once the scale-space is computed, they use the determinant of the Hessian to locate image feature point candidates, and prune some features which are unlikely to contribute to the matching process.

The results presented show that their method outperforms SIFT and SURF significantly in six image pairs. The prose suggests a larger number of tests might have been performed, and that they used variable amounts of motion blur. Results showing variation in performance with motion blur length or direction were not given.

Other results present the number of correct matches between images which have been warped using planar homographies computed using their matches. A threshold of a small number of pixels is used to determine whether feature pairs match or not. Their method does not produce matches when the threshold is very strict — large numbers of matches seem to be produced with a threshold of two pixels or greater. As noted above, the direction sensitivity of SIFT descriptors is particularly prone to suffering mis-matches due to motion blur. The alterations to scale space described by the authors should be expected to produce an improvement over SIFT or SURF. The processing time of their implementation is not immediately suitable for real time applications: They report an execution time of 1 second per 640x480 frame, therefore this is an offline method.

2.9 Motion Blur Removal

The main objective of this Thesis is to match features between an image with motion blur and one without, at video frame rate. A straightforward approach to this problem might be to take an existing method for removing motion blur and apply it to the blurred image. If the motion

blur was successfully removed, then many of the existing feature matching methods might be suitable.

This section of the review examines some recent developments in the removal of motion blur. Methods are examined based on their computation time, and the quality of their results. Whilst computation time can be examined objectively, none of the resulting images are tested in a feature matching system. As will be seen, the computation times of all methods are far too long, so the ability of the de-blurred images to be matched is not relevant.

State of the art image de-blurring methods fall into one of threee broad categories; deconvolution, multi-image, and data driven methods which use Convolutional Neural Networks (CNNs).

2.9.1 Deconvolution

Deconvolution could be considered the "classical" signal-processing approach to de-blurring. If the image is treated as a signal on two-dimensional support $I(\mathbf{x})$, as described in Section 3.2.3, then motion blur can be expressed as a convolution

$$I(\mathbf{x}) = k(\mathbf{x}) * I_s(\mathbf{x}) + n(\mathbf{x}) \tag{2.2}$$

where the symbols have the same meaning as in Section 3.2.3. The problem of de-blurring is then to find I_s given I. This is an ill-posed problem. There are two possible avenues to solving this problem. One is to attempt to estimate the blur kernel k from I, then deconvolve I_c from I and k. The other is to attempt to find I_s and k simultaneously from I. These are referred to in the literature as "non-blind" and "blind" respectively.

Non-blind Deconvolution

The "non-blind" collection of methods are rather varied. Various approaches are taken to estimating the blur kernel. A number of different options are available to carry out the deconvolution once the blur kernel has been estimated. The "Richarson-Lucy" algorithm is cited as the basis of this approach to de-blurring by a number of papers . Two papers by these authors [55, 56] proposed the idea of treating the restoration of an image with a known filter as a Bayesian statistical problem. More recent methods manipulate the Prior probabilities used to control the sharp image as it is being computed, and the optimisation technique used to find the sharp image. The well-known Wiener filter is another common method for blind deconvolution of signals corrupted by noise. Of the methods discovered during this review, only one uses Wiener filtering to deconvolve the latent image from the blurred image and the filter, once the filter has been found.

Zhang and Hirakawa [57] model the kernel using the Dual Discrete Wavelet Transform. Their results show ghosting and ringing artefacts, so the method is probably not suitable for feature matching. They don't mention execution time. Zoran and Weiss [58] propose attempting to learn priors on image patches. Patch-based restoration is used to attempt to lighten the computational load compared to whole-image restoration. Learning the statistics of patches seems more tracable, and likely to be useful, than statistics over whole images. They show promising results on de-noising problems with results similar to other state of the art methods. The run-time is 300s per image, which is too slow for real-time applications.

Shah and Dalal [59] use a method related to the Cepstrum to find the length and orientation of an in-plane motion blur filter. They show that by taking the fourth bit-plane of the log-spectrum-log-specrtum of a blurred image, then taking the Hough transform, the parameters of the blur can be found. The sharp image is then found using Wiener filtering. In their test data, the upper right and lower left triangular segments of a rectangular image are blurred with different motion blur filters. The results of their method are very crisp and are able to restore much detail, but they are corrupted with artefacts which appear as diagonal lines on the image. It isn't clear if these artefacts are a consequence of the test set-up, or the method. These artefacts are likely to appear as false positives or noise in a feature matching algorithm. Run-time is not discussed.

Blind Deconvolution

Blind deconvolution effort appears to be focussed on the refinement of a Bayesian approach. Fish et al [60] proposed the original extension of the Richardson-Lucy algoriothm to blind deconvolution problem.

The probability maximisation takes the form:

$$\arg\max_{I_c,k} p(I_c, k|I) = \arg\max_{I_c,k} p(I|I_c, k)p(I_c)p(k) \tag{2.3}$$

Here $p(I|I_c, k)$ models the noise, $p(I_c)$ models the distribution of possible sharp images, and $p(k)$ models the distribution of blur kernels. The left hand side says we want to find the values of I_s and k which are most likely, given I. This equation can be rewritten[61] as :

$$\arg\min_{I_c,k} \|k * I_c - I\|_2^2 + \lambda J(I_c) + \gamma G(k) \tag{2.4}$$

here, $\|\cdot\|_2$ is the L2 norm. The first term is the data fitting term, which describes how well the convoluved k and I_c match f. The other two terms are regularization, which control how the minimization can vary I_c and k. This formulation invites experimentation upon choices of J, G, λ and γ, as well as approaches to optimisation.

The early approach due to Fish et al [60] was to alternate between optimisations to find I_c, assuming k was known, and finding k assuming known I_c. This was a direct extension of the Richardson-Lucy algorithm to find k as well as I_c by assuming the current value of one was correct, and minimising the cost to find the other, then alternating.

A modern approach which scores well in comparisons is "Total Variation". This method was first proposed for blind deconvolution by Chan and Wong [62], as a modification of a non-blind method. Perrone and Favaro [61] present a clear description of Total Variation, with a discussion clarifying some formerly contentious points. In this paper, they set $J(I_c)$ to be the L1 norm of the L2 norm of the gradient of I_c, which is the same total variation regularizer used by Chan and Wong. They set γ to 0, and instead control the behaviour of k by applying constraints on that it must be positive everywhere and normalized. (The contentious point clarified by Perrone and Favaro is that applying these constraints sequentially instead of together influences the behaviour of the algorithm such that it is no longer a pure gradient descent method, and can escape local minima.)

Perrone and Favaro [61] show that by carefully selecting λ their simplified method produces results of similar quality to the modern, sophisticated implementations of Total Variation. Their implementation takes between 2 and 5 minutes to compute a 255 pixel square image. This is far too slow for video rate processing, and even with aggresive optimisation it seems unlikely to be suitable for real-time processing. The examples of restored images provided in the paper are of relatively high quality. Their are some artefacts, and softness to the images. But the motion blur is almost completely gone.

Takeda and Milanfar [63] proposed a blind method for removing blur from one frame of a video sequence. They assume motion in the video is locally smooth in space and time, and then upsample the video in both space and time. I_c and k are then found at the original resolution by applying a Bayesian energy minimisation. The results they have chosen for visual comparison contain very little motion blur. The results of this method appear visibly sharper than the comparison, but not significantly so. They don't mention the execution time, but the method includes two optimisation steps. It's quite likely this will take considerably longer than a video frame period, so this method is likely unsuitable for real-time tracking.

2.9.2 Multi-image

Multi-image methods reconstruct a non-blurred frame from a number of input frames. The assumption here is that the objects in the blurred frame will appear somewhere in the one of the other frames either without blur, or with sufficiently different blur to enable a non-blurred image of the object to be constucted. There are fewer of these methods in the literature, and

they are not necessarily appropriate for the broadcast camera tracking case, as the camera motion (and resultant blur) tends to be similar in adjacent frames.

Delbracio and Sapiro [64] register images together using filtered SIFT feature matching. (Note that Gauglitz [1] found the Difference of Gaussian detector, used by SIFT, to have poor repeatability under small amounts of motion blur, when comparing images with differing blurs. This means the registration process will be error-prone) then accumulate the Fourier transform of the images, and reconstruct an image by taking the inverse Fourier Transform. The visualised results show considerably reduced noise compared to the input, and mostly sharper images. However, occasional spurious features appear in some images, as some part of the image is reconstructed in the wrong place, or duplicated. The run-time is reported to be "a few seconds".

Considering use in a real-time tracking system, this method has two properties which make it unsuitable: First, it requires multiple images as input, each with a different, random blur kernel. Secondly, the run time is too long.

2.9.3 Data-driven

CNNs have been applied widely to computer vision problems with great success over the last few years. Since demonstrations of style transfer [65] and the unique results of the "deep dream" experiments, it was made clear that CNNs were capable of synthesising images in a convincing way.

An Artifical Neural Network (ANN) is a method for computing a mapping between an input and output. The input usually contains many more values than the output. The architecture of an ANN is modelled after the connections of the mammalian brain. Each neuron takes a weighted sum of its many inputs, applies a non-linear activation function, and returns an output. Neurons are typically arranged in layers where each neuron will have the same inputs as its layer-mates, but will each have different weightings. A schematic artifical neuron is shown in Figure 2.15 and a simple ANN is shown in Figure 2.16

[1]Figure due to Gonzalo Medina, from `https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network`

[2]Figure due to Gonzalo Medina, from `https://tex.stackexchange.com/questions/132444/diagram-of-an-artificial-neural-network`

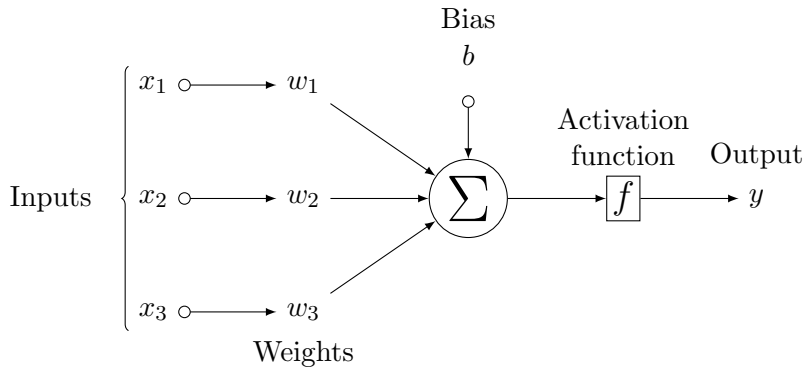


Figure 2.15: An Artifical Neuron [1]

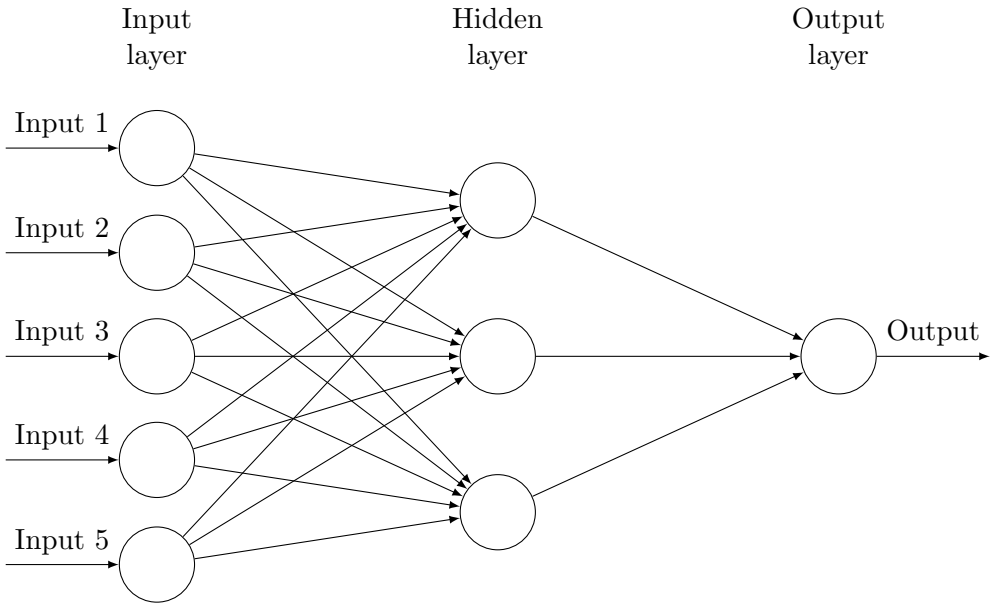


Figure 2.16: An Artificial Neural Network[2]

A training phase is used to set the values of the weightings using a technique called back propagation. A data set of ground truth inputs and outputs is required. An input is presented to the network, and an error is computed by comparing the output with the ground truth. This error is then used to modify the weights at each node in a way proportional to the gradient.

If the training has been successful, the ANN can now be used on an unseen input to discover the corresponding output. Training success depends heavily on the quality of the training data.

A Convolutional Neural Network (CNN) is a modification of the ANN where a single neuron can take as input a subset of the input signal, rather than the whole signal. This means that data relating to different parts of the signal propagate through different parts of the network. CNNs also tend to be much larger than ANNs, A complete CNN will usually include so-called "fully connected layers" which are configured like ANNs. CNNs have been found to be capable of creating images which are realistic [66], (although they can also create images which are not realistic, but are fascinating [67].) This ability to synthesise realistic images has lead some researchers to investigate how well they deblur images.

A few papers have attempted to use CNNs to remove motion blur from an image, either directly or indirectly. Su et al [3] describe an architecture where the conventional CNN structure is augmented with a second network, reversed. Figure 2.17 shows a diagram of the network. (For definitions of the different types of layer, see their paper.) There are three "skip connections" which enable nodes in the second network to access information in the first. This network is trained using blurred image sequences as input and the central unblurred image as output.

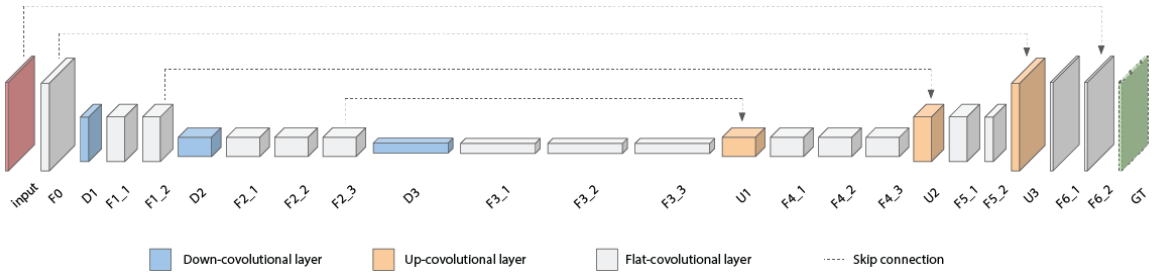


Figure 2.17: The network architecture used by Su et al [3]

The method can incorporate side-channel data such as inter-frame homographies to enhance performance. The visualised results seems somewhat content dependant, and are clearly improved when the homographies relating the input frames, or the optical flow, are included. The run time reported is around one second per frame. This is too slow for real-time visual tracking, and the visual quality might cause problems for tracking systems. The main drawback with this method is the requirement for optical flow, or homographies, for the highest quality result. Computing this data on blurry sequences is hard to do, and not possible to do

quickly (That being the central question of this Thesis!)

Noroozi et al [68] use a CNN to find a residual which is added to the blurry image to give the sharp version. They also provide a data set made up of high frame rate video, from which blurry frames are synthesised by frame averaging. In a similar manner to Su et al [3] the network architecture starts with image-scale convolutional layers, before reducing in size, and later increasing again. The visualised results are content-dependent. In some cases, the motion blur is only reduced rather than eliminated. In other cases, artefacts which look like mis-placed image content are apparent. They do not mention the run time to synthesise a sharp image from a blurry one. The network architecture and scale is not too dissimilar from Su et al [3], so a similar run time of around 1 second seems likely.

Chakrabarti et al [69] propose a CNN based approach which incorporates more domain-specific knowledge. The input to the network is a custom frequency domain representation of an image patch, and the output is the deconvolution filter required to find the unblurred patch from the blurred one. The network is trained on synthetic data; sharp images are blurred with a synthetic blur kernel. The network works patch-wise, on dense, overlapping patches. An optimization process recombines the unblurred patches into a sharp image. Two sets of results are given as visualisations. The first set use images corrupted with very heavy blur and noise as input, and produce results which appear to contain significantly more detail than the input. The second set contain less blur and noise, but show a wider variety of scenes. The ability of this method to remove blur and not produce artefacts seems to be somewhat content-dependant, based on this set of results. The methods chosen for comparison also show a similar variability. The execution time they report is 65s. This is too slow for consideration in a real-time system.

2.9.4 Conclusion

Removing motion blur from an image, with no other information, is a challenging task. The papers surveyed contain a wide variety of approaches, representing the state of the art, as far as the author is able to determine. All approaches have significant drawbacks for the task of real-time visual tracking. Of the methods which produce results which appear relatively good quality they all either take significantly longer than a video from to compute [58, 61, 3, 69] or don't report their run time, but contain tasks which are conventionally computationally slow, such as large non-convex optimizations[57, 59, 63, 68]. Delbracio and Sapiro [64] takes too long, and also needs images in a format which would not be available from a tracking broadcast camera.

It seems that, at present, de-blurring a blurred image is not a sensible approach to real-time visual tracking of blurred and non-blurred images.

2.10 Optical Flow from Motion Blur

A number of methods exists to compute optical flow in the presense of motion blur. Whilst optical flow is not the main focus of this Thesis, it is a related area, and worth examining briefly in case there are techniques which are suitable for inclusion in these experiments.

Most of these rely on the assumption that the motion of an object between a pair of frames will be closely related to the amount of motion blur on that object. Portz et al [70], Kim et al [71], and Daraei [72] all provide energy functions which incorporate the formation of blur from a latent sharp image and some spatially-varying motion. Portz at al [70] rely on the commutative property used by Jin et al. They estimate knowledge of the unknown blur by computing a few hundred different blurs with different lengths and orientations, sampling and interpolating as needed. This method requires lots of computation time and memory. The cost function of Kim et al [71] uses the optical flow estimated on the latent images along with the latent images to measure similarity to the observed blurry images. This is solved in a large optimization. The authors do not remark on execution time except to say the current implementation "is time consuming and needs large resources". Daraei [72] estimates the motion blur on the latent images by warping one set of optical flow using the previous and next optical flows. The visualised results and the numerical results are very good compared to the competition, but still high frequency lack detail. Again, there's no mention of execution time, but a large scale optimisation is required, so this method is expeced to be too slow.

Schoueri et al [73] claim to compute optical flow, but actully compute a 2-parameter motion blur kernel for a a grid of image tiles. Tu et al [74] modify the existing non-blur-aware method of Brox et al. They use a mask so that their deblurring operation only occurs on parts of images detected to be blurred. They also add an edge-preserving regularization. The blur detection and labelling part of the algorithm along takes 113 seconds for one image. This is too slow for real-time use.

Li et al [75] attach a motion sensor to the camera and incorporate the sensor data into the optical flow measurements. The pre-processing combines methods from several previous methods to improve the results. The visualised results appear good, and mostly free from artefacts. The computation takes tens of seconds per frame which is too slow for real-time use.

All of the methods found are too slow for real time use. Even the addition of a hardware sensor, which is explicitly beyond the scope of this Thesis, does not improve run-time sufficiently to be useful in this context.

2.11 Feature Matching incorporating de-blurring

This topic does not appear to be heavily investigated, as far as could be determined. Certainly as the main topic of reported work, the approach of deblurring before performing matching is not common. This is the only paper on the topic discovered during the literature review:

Deblurred SIFT

Okade et al [76] describe a method to improve video stabilization by visual tracking in the presence of motion blur by deblurring blurred video frames, then applying SIFT feature matching. They use the integrating camera model described in Section 3.2.3. They detect blurred video frames by taking the sum of the squared image gradient in both x and y, and comparing it to a threshold, set per-sequence. Those frames found to contain blur are deblurred using a method due to Fergus [77]. Then SIFT features are matched between the newly deblurred frame and an earlier frame, and used to estimate the inter-frame motion. The inter-frame motion is smoothed with a low-pass filter.

Attempting to recover a non-blurred version of a motion blurred image is error prone. The motion blur filter contains zeros, and small coefficients, which mix the signal with thermal noise in the cameras. The results shown in [77] are subjectively of good quality, but still contain some artefacts which appear similar to ringing, or ghost images.

Their results show an improvement in Interframe Transform Fidelity (ITF), their chosen metric, for three sequences. There are no results showing detailed analysis of performance with respect to a controlled variation of motion blur parameters. As their application was video stabilisation, some subjective measurements of subjective video quality would have been informative. Given the good subjective quality of the results in [77], some detailed analysis of the impact of the few remaining artefacts upon the matching process would have been very useful.

2.12 Summary

Visual tracking in the presence of motion blur is addressed sparsely in the literature. Of the papers [43, 46, 44, 78, 2, 76, 49, 53, 54, 36] which address the problem, none attempt to assess their methods using the standard methods introduced by Mikolajczyk et al [27, 4]. The only review paper [1] which does present any formal analysis of visual tracking performance in the presence of motion blur does not include any of these specialised methods.

Of particular interest to this Thesis are methods which can deal with planar motion blur which differs between the two image regions being considered. The method should be com-

putationally efficient, ideally suitable for inclusion in a real time system, as described in Section 2.1.1.

Several of the detection, template matching, and descriptor methods were categorised as real time. To review, they were:

Table 2.1: Summary of real time methods

Detectors	Template Matchers	Descriptors
Harris	Phase Correlation	SIFT
Difference of Gaussians	KLT	GLOH
Fast Hessian	ESM + Pixel-wise	PCA-SIFT
MSER	Simultaneous Minimisation	SIFER
		SURF
		Phase-based Local
		Fourier Phase Quantisation

Of these methods, none explicitly addressed the problem of matching features between images containing motion blur and images without motion blur. The analysis of Gauglitz et al [1] finds that performance falls off rapidly with increasing blur length for SIFT and SURF. The problem addressed by this Thesis is an appropriate one, then. There is little work addressing the specific problem, and the methods which have been tested do not perform well.

Of these real time methods, which should be selected to be modified in an attempt to make them more robust to differences in motion blur between images? A straightforward relationship between motion blur and some property of the algorithm is important, as it means modifications are readily identifiable and testable. The faster the method, the better, as the extra work to deal with motion blur is likely to incur some additional computation. For these experiments, we are concerned with pan-tilt-zoom cameras and stationary backgrounds, as described in Sections 1.1.1 and 1.1.2. So sophisticated methods which can model full camera motion and pixel-wise varying blur are too general, and will likely take up more computational resources than necessary.

The Harris detector is used throughout these experiments, as it is used in the Piero system. The experiments later will then be a good predictor of performance of these methods were they to be incorporated into Piero. As noted later in Chapter 4, the Harris detector also seems like the best choice for comparison with the results of [1].

To test modification to template-matching, Phase Correlation is selected. Phase correlation is chosen over the KLT because of the clear mapping between coefficient position and motion blur. In addition, Dawes et al found [14] that the KLT is efficient when motion blur variation between frames in small, but cannot handle large variations in motion blur, which can be important for preventing drift. Of the other real time matching methods, ESM with pixel-wise

blur estimation [42, 43] already attempts to track features in the presence of motion blur. To run in real time the method runs in real time and is capable for solving for pixel-wise motion blur and a fully moving camera. This is too general a method for the problem investigated in this Thesis.

SIFT features are also chosen for their clear mapping of direction onto coefficient, which allows reasoning about how each coefficient might behave under motion blur. SURF [30] would also have been suitable for analysis. The more straightforward modifications to SIFT were performed first, and time was not left after the SIFT experiments to investigate SURF also. Similar arguments could be made for using Phase-based local features [52]. Fourier Phase quantisation [53] already attempts to deal with motion blur, and performs poorly, on short motion blurs. GLOH [4] and PCA-SIFT [47] are not suitable because the relationship between motion blur and coefficient is masked by the PCA operations. SIFER [48] is noted by the authors of the paper as beign susceptible to differences in motion blur.

The review of the literature on deblurring given in Section 2.9 that there are no real time methods for deblurring which approach state of the art performance. Indeed, there is no recent work on real time blind deblurring. This finding means that the approach of deblurring the blurred frame and matching to a frame with no blur is not suitable when the matching algorithm is required to run in real time.

Chapter 3

Experimental Validation of Motion Blur Model

3.1 Introduction

This Chapter described the investigation into question 1 of this Thesis: "Does motion blur behave as conventionally modelled?" This contribution consists of an experimental design which both provides evidence supporting the conventional model, but also is sensitive to noise and low spectral occupancy in images in a way that makes it useful as an assessment of camera quality. The experimental design requires very little specialist equipment — no frequency-sweep test cards or calibrated lighting. As such it is a more accessible method for assessing camera quality, when noise and sharpness are the camera qualities of interest.

The models for motion blur described under the integrating camera in Section 3.2 have not been experimentally verified in published work known to the author. In this chapter, an experiment and results are presented which verify the model of motion blur as a rectangular filter. It is shown that, for a moving edge, the experiment also verifies the model of the pixel as an integrator.

The experiments in this Chapter were designed to verify that there were no deviations from the integrating camera model in a range of professional broadcast and feature film digital cameras. Manufacturers are sometimes not willing to disclose details of how their cameras work, and usually for sensible commercial grounds. Commercial grounds are of no use to an investigator who wishes to discover precise details, however!

In which ways might a video camera deviate from a perfect theoretical model?

- Does the proportion of incident charge accumulated vary under different intensity, already accumulated charge, or anything else?

- Does charge leak between adjacent photo-sites?
- Electronic shutter rise and fall times are modelled to be instantaneous. Is this the case?
- Is the precision of exposure duration set by frame rate and shutter sufficiently high?
- Are the unexpected non-linearities anywhere else in the system?

The experiments described below were an attempt to examine these effects and how they might influence the formation of motion blur. Deviations found can be taken into account when attempting to design visual tracking systems which attempt to account for motion blur.

This Chapter is structured as follows: In Section 3.2 the instantaneous and integrating camera models introduced in the Literature Review are expanded upon. In Section 3.3 the model of motion blur is described in more detail, with some verification based on idealised theoretical camera behaviour. Then in Section 3.4 an experimental method is described to measure the response of several cameras to a test signal designed to generate highly reproducible motion blur. Section 3.5 describes the results of the experiment, which are discussed in Section 3.6. The Chapter is concluded in Section 3.7.

3.2 Camera Models

Two distinct camera models are used in visual tracking. The *instantaneous camera* models stationary scenes, and thereby enables tractable mathematical analysis [79]. The *integrating camera* introduces relative motion between the camera and the objects in the scene, thereby providing a framework for dealing with motion blur. The following two sections describe these camera models, and list the image distortions each can account for, when trying to find correspondences between pairs of images.

3.2.1 Instantaneous Camera

Most published algorithms [38, 10, 31, 80, 20, 30, 81, 40, 4] rely on an instantaneous camera model. This model can account for changes in scene brightness, contrast, gamma correction, viewpoint, object scale, central blur, and in-camera processing. Under certain conditions this model is accurate enough for reliable tracking.

In the instantaneous camera the entire field of view is exposed at the same instant. Light is modelled as a classical ray, and travels instantaneously. Neither the camera nor objects in the world are assumed to move during the exposure.

Geometry

An instantaneous camera is modelled by a focal point, focal plane, angle of view, and orientation. A point in 3D space $\mathbf{X}$ is projected into its image $\mathbf{x}$ on the focal plane of the camera as

$$\mathbf{x} = P\mathbf{X}$$

where P is the 3×4 projection matrix. The projection matrix encodes all the intrinsic parameters of the camera, as well as its position in space and orientation. Hartley and Zisserman [79] introduce this model very thoroughly, but briefly, the projection matrix can be broken down into

$$P = K[R| - R\tilde{\mathbf{C}}]$$

where K is the camera calibration matrix, R is a rotation matrix describing the orientation of the camera in the world reference frame, and $\tilde{\mathbf{C}}$ is the position of the focal point in the world reference frame. Figure 3.1 below illustrates the action of a camera matrix incorporating a rotation matrix $R = I$. K can be decomposed into the individual parameters intrinsic to the camera:

$$K = \begin{bmatrix} f & s & c_x \\ 0 & f/r & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$

f is the focal length and r is the aspect ratio. c_x and c_y are the coordinates of the point where the camera direction of view vector intersects the focal plane, in the focal plane coordinate system, and s describes skew, which is usually zero.

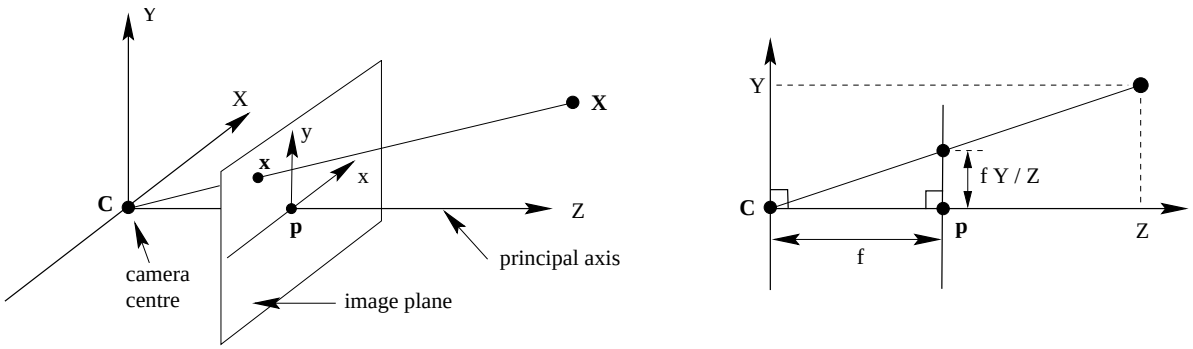


Figure 3.1: On the left, the 3D point $\mathbf{X}$ is transformed into the image coordinate $\mathbf{x} = P\mathbf{X}$. The right figure shows the relationship between a 3D coordinate, its image, and the focal length.

Spatial Quantisation

In a real camera, the focal plane is quantized into regions which integrate the incident light to produce an output signal. In digital cameras this quantisation is a regular grid. Broadcast and motion picture cameras typically output three channels at each of these photo-sites, one each for red, green, and blue. In some cameras this data has been captured directly by three sensors, behind filters. In others, the output is interpolated from a single sensor, whose individual photo-sites are sensitive to different colours.

Depending on the camera, a number of linear and non-linear steps might be taken between the voltage output by the sensor and the digital output image. Their effects can be seen in the output frames, but the parameters and operations used to create them are usually masked from the user. Thermal processes within the sensor also produce some random variation in the pixel values, which can only be modelled statistically as an unknown noise. This camera model results in a three-dimensional array of values, which represent the digital image.

3.2.2 Instantaneous Camera: Modelled Distortions

Image distortion can be broadly defined to mean any change in the pixel values representing the image of a particular object. The resulting image may be more or less recognisable, subjectively. But this Thesis is concerned with algorithms which may be sensitive to properties of images which the human visual system is not. So this broad definition of image distortion will be used. The instantaneous camera is capable of modelling any per-pixel image distortions, and any distortions which occur at the level of the image as a sample array. Changes in appearance resulting from relative changes of the position of objects *between* frames can be modelled.

Per-Pixel Distortions

Given a pair of images from cameras in identical positions, pixels observing identical parts of the real world can still differ in their output value. Brightness, contrast, gamma and colourspace model the mapping function between two such pixels.

Colourspace and gamma for an input image are usually known, as they are set by the camera operator. Brightness and contrast can be influenced by the camera iris, exposure, optical filtering, and electronic gain and bias. (They will also change if the illumination changes.) All of these changes map single pixel values between images.

Gamma correction is used in every television and video frame. It is a nonlinear mapping which has multiple purposes in data reduction and psychovisual accuracy. Sample values must be converted from gamma corrected to linear values if they are to be processed accurately.

In the EBU Specification for HD TV [82], gamma correction is to be performed by calculating

$$I_{out}(\mathbf{x}) = (1 + a)I_{in}(\mathbf{x})^{\gamma} - a$$

where I_{in} and I_{out} are the normalised channel luminance values before and after gamma, $\mathbf{x}$ is the pixel coordinate, γ is the gamma correction coefficient, and a is set to 0.099, for HD TV. All cameras used for TV production use equations based on this, sometimes with additional adjustments to allow more control over the look of the image. Parameters are either as defined in the specification, or closely related ones. Digital Film cameras often use considerably different curves [83].

If it was desired to undo the effect of gamma correction, if would be sufficient to apply the inverse of the above equation. This is rarely done, however: Of the papers cited in this review, [77] mentions that gamma correction is removed as part of the method, the review paper, [27], notes that one method is relatively immune to gamma, [53] notes in their conclusion that one of *their* references removed gamma and found improved results. [50] apply additional gamma to their signal before analysis. None of the other papers reviewed mention gamma.

It is perhaps worth considering the assumption of constant brightness is a sensible one for visual tracking, and industrial applications. In that case, correcting for gamma would change the inter-frame match only a little, or not at all. Many of the methods described below claim invariance to illumination changes. These methods would need to model gamma somehow, or suffer some additional error.

Changes in colourspace can be modelled trivially if the colourspaces of both images are known. The primaries for colour spaces are clearly defined, and converting between them usually just requires a 3×3 matrix multiplication of the colour channels. Broadcast, motion picture, and still cameras all output images in a colourspace which is convertable to RGB.

Most approaches to visual tracking operate on luminance. When a single camera is involved, any linear combination of the colour channels would be satisfactory. When trying to identify features or match regions between different cameras, care must be taken to extract signals from the cameras which are equivalent. For example, by finding the matrices to convert from the cameras colourspace to a defined luminance.

Brightness and contrast can model the remaining difference in pixel values between the images. In the earlier example with two identical cameras observing the same scene from the same position, once gamma has been removed and colorspace matched, we expect to be able to model the pixels in one image I_1 as a function of the pixels in the other image I_0:

$$I_1(\mathbf{x}) = b + cI_0(\mathbf{x})$$

where b is the brightness and c the contrast.

Affine Distortion

Differences in image viewpoint and scale can be modelled as affine distortions of the image space. The affine camera is used as an approximation to a perspective camera [84, 79], and certain constraints apply for the approximation to be valid: The largest distance in world space between the points subject to the transform must be small compared to the distance from the camera centre to the points. Under these constraints one image can be represented as an affine warping of another by

$$I'(\mathbf{x}) = I(A\mathbf{x})$$

where A is a 3×3 matrix, and the image coordinates $\mathbf{x}$ are expressed in homogeneous coordinates. In general A has 8 degrees of freedom; matrices which are related by a constant factor produce identical transformations.

Image-level distortions

Two further distortions can be modelled straightforwardly: *In-plane rotation* is modelled as a transformation of pixel coordinates. *Defocus blur* can sometimes be modelled using a filtering operation applied to a notional unaffected image. Different lenses have different point spread functions, and some will be more straightforward to model than others.

Other Distortions

Some distortions are more difficult to model accurately, and can be treated as sources of error:

Noise. All images contain some noise. Some high quality cameras include a measurement of the fixed-pattern noise, which is made at the time of manufacture, which is subtracted from every frame. But random noise from thermal processes is still present. In analysis algorithms, this is modelled as a small random error.

Obfuscated In-Camera Processing. Broadcast cameras include features with names like "Edge Enhancement", "Aperture Correction", and "Sharpness", which usually have one or two parameters which can be controlled by the user. Their effects are never explicitly described in operators manuals. These features can sometimes be disabled. When present, they must be taken into account as a source of unknown, nonlinear error.

3.2.3 Integrating Camera

The integrating camera is an extension of the instantaneous camera. It is not well defined in the literature. Here it is an umbrella term for models which assume there is an underlying motion between the objects in the scene and the camera during a single exposure.

Incorporating all of the factors which influence the appearance of a picture into an analysis tool would be complicated. The feature film industry is getting quite close to producing convincing synthetic reality, including diffraction effects from particles, specular reflections, sub-surface scattering for convincing human skin, and so on. This shows that the models are rather accomplished on the synthesis side. Still, incorporating such detail into analysis, particularly in real time, is beyond the state of the art today. Further, the feature film industry has shown that the benefit of adding refinement to physical models diminishes as more and more refinements are added.

A number of papers [42, 43, 45, 44] have made steps towards a more realistic model by incorporating some dynamics into their camera models. Some model global motion, others incorporate per-pixel relative motion between the camera and scene. All assume that the colour of a region in the real world will not change as it moves. Specular reflections, and other physical effects which cause significant changes in brightness over the small motions which occur during a frame are considered to be a source of noise.

3.2.4 Integrating Camera: Modelled Distortions

Incorporating dynamics into the instantaneous camera model allows motion blur and rolling shutter distortion to be modelled. No standard model for motion blur exists; the papers cited in the previous paragraph all use related but different models.

Motion blur

A common approach to motion blur is to model it as a filter. The motion blurred image $I(\mathbf{x})$ is modelled as a notional instantaneous image $I_s(\mathbf{x})$ convolved with a rectangular filter $k_{\mathbf{v}}(x)$. The subscript $_{\mathbf{v}}$ indicates k is parameterized by a motion velocity in image space with orientation θ and length L. This model assumes the motion can be approximated by a straight line in the image plane. The length L will be a function of the relative motion between the camera and the scene, the frame rate, and the shutter setting of the camera. L will be in inverse proportion to the frame rate, and direct proportion to the shutter.

$$k_{\mathbf{v}}(x) = \begin{cases} 1/L & \text{if } -L/2 < x < L/2 \\ 0 & \text{otherwise.} \end{cases}$$

In two dimensions the rectangular filter kernel is a straight line one pixel wide and L pixels long. The length of the blur and angle of the line to the sample structure depends upon the relative motion between the camera and the object. The filter kernel extent and angle can vary across the image, representing non-uniform motion indicated by an additional subscript $\mathbf{x}$. A motion blurred image I would then be modelled as follows:

$$I(\mathbf{x}) = k_{\mathbf{v},\mathbf{x}}(\mathbf{x}) * I_s(\mathbf{x}) + n(\mathbf{x})$$

where $*$ is the convolution operation and $n(\mathbf{x})$ is camera thermal noise.

Clearly this model cannot accurately describe image regions which are revealed or occluded as part of the camera-scene motion. Also, the modelled motion must have constant speed (owing to the flat, symmetrical filter kernel). More sophisticated models, discussed later, allow the parameters of the motion blur filter to vary over the image, or allow non-straight lines as the motion blur kernel.

Phase Inversion The Fourier transform of the rectangular filter is the sinc function. The sinc function has negative lobes for certain frequencies, which can be thought of as inverting the phase of those frequencies. This interpretation of the effect of motion blur is used throughout this Thesis.

Schelten and Roth explored [85] the removal of motion blur by estimating a blur kernel like $k_{\mathbf{v},\mathbf{x}}$ in a free, non-parametric way, so as to estimate blur arising from non-linear (in time and space) motion. Their method has not been applied to tracking in the literature.

Additional models of motion blur used in different integrating camera models will be introduced along with the papers describing them.

Rolling Shutter

The instantaneous camera model implies a "global" shutter — The entire frame is exposed at one instant. CMOS sensors, which are now widely used in cameras, violate this model. The sensor is read out one line at a time, so each line is integrating the scene for a different period of time. The amount of distortion introduced is proportional to the amount of time which elapses between the first line starting its exposure and the last line. (It also depends on the horizontal motion speed of objects in the scene.) Some sensors mitigate this by reading out into a buffer more than once per frame, so that the total time offset between the first and last line is reduced from the frame time to an integer fraction of the frame time.

Models for dealing with rolling shutter distortions are still a subject of research. An integrating camera model could potentially deal with rolling shutter distortion in a more sophisticated way than treating it as a source of uncertainty. This review does not look at rolling shutter effects in depth. The models for motion blur described under the integrating camera in Section 3.2 have not been experimentally verified in published work known to the author. In this chapter, an experiment and results are presented which verify the model of motion blur as a rectangular filter. It is shown that, for a moving edge, the experiment also verifies the model of the pixel as an integrator.

3.3 A model of motion blur

In this section a model of motion blur is described using standard approaches and simplifications, which place some restrictions on the experimental design. A type of integrating camera is modelled, observing a planar scene.

3.3.1 The pinhole camera

The pinhole camera is a model of a lens and sensor as a focal point and a focal plane respectively. Rays from the world are traced through the focal point until they intersect the focal plane. An object moving parallel to the focal plane with uniform speed will form an image on the focal plane moving at uniform speed, by similar triangles.

This model has some shortcomings when dealing with real lenses. Real lenses deviate from the model with barrel distortion, tangential distortion, chromatic aberration, and depth of field. Barrel distortion, tangential distortion, and chromatic aberration can be dealt with to a high degree of accuracy by using a high quality lens. The experimental design in Section 3.4 includes means to deal with depth of field, and further ameliorate the effect of barrel distortion.

The pinhole camera defines how real objects are imaged on the focal plane. In a real camera, this image must be sampled.

3.3.2 Integration and sampling

The idealised image sensor is made up of a number of light-sensitive photo-sites, which accumulate charge proportionally to the number of photons which fall on them. The output of these photo-sites are processed into an output array of pixels. Even with the camera parameters fixed, some video cameras perform non-linear functions to compute the pixel values, sometimes these functions depend upon more than one photo-site. The most significant of these in the material gathered for the experiments described in this Chapter are gamma and edge-enhancement. Gamma does not have any spatial component. The output value of the gamma stage for the pixel at $\mathbf{x}$ is dependent only on the input at $\mathbf{x}$, and the gamma parameter. It can be corrected for if the mapping is known.

Edge enhancement is a filtering process and hence combines nearby pixels to find the output at $\mathbf{x}$. It is a high-pass filter, which only effects sharp edges; since here we deal with blurred edges, the effect of edge enhancement can be safely ignored. Other processing like noise reduction will be a source of uncertainty in these experiments.

For the purposes of this model, the value of the pixel $I(\mathbf{x})$ output by the camera is proportional to the charge $V_{out}(\mathbf{x})$ accumulated by the sensor at $\mathbf{x}$: $I(\mathbf{x}) = qV_{out}(\mathbf{x})$.

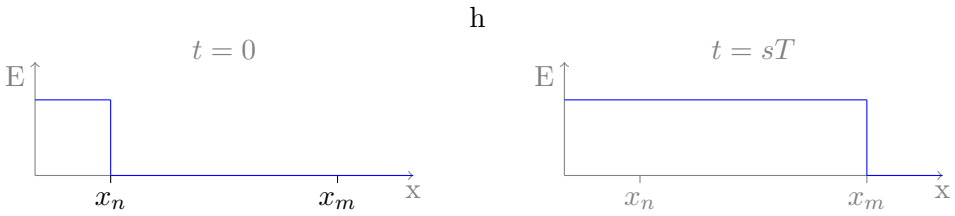


Figure 3.2: The motion of the intensity edge through during the exposure

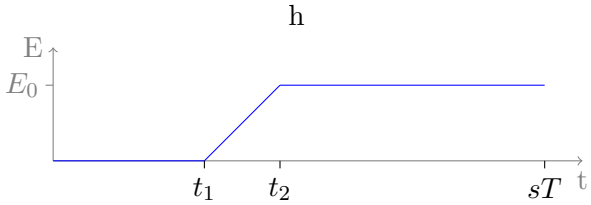


Figure 3.3: The intensity function at the pixel $x_{n+\Delta n}$

Each photo-site has a finite size, and so will accumulate together small details. An optical low-pass filter is present in front of the sensor on most cameras to prevent aliasing. Moving shapes will be distorted somewhat by the sampling structure, depending on the relative size of the motion on the sensor and the photo-site size.

3.3.3 A moving edge

In these experiments, motion blur is measured using a moving edge. A high contrast straight edge moving with constant speed in the direction of maximum gradient is a useful tool for measuring motion blur. The constant motion of the edge, and constant sensor illumination either side, result in a straight line blur. This makes for relatively straightforward measurement. The simple framework described below shows how a straight line blur arises from constant motion of a straight edge without considering filtering:

A moving edge is a straight line illuminated to one side, moving perpendicularly. A pixel on the illuminated side is subject to an intensity E, and a pixel on the dark side receives 0. The sensor has square pixels p_x metres across. The edge is moving at a speed v ms^{-1} across the sensor such that at the beginning of the exposure the edge is at column n and at the end of the exposure is at column m (Figure 3.2). For simplicity the start of the exposure is defined to be $t = 0$, and the end of the exposure to be sT, where T is the frame period in seconds and s is the shutter factor, expressed as a value between 0 and 1. A constant level of illumination E_0 is chosen such that

$$\int_0^{sT} E_0 \, \mathrm{d}t = 1$$

Based on this model, the intensity function at some pixel $n + \Delta n$ must vary over time (Figure 3.3). Integrating this intensity function will give an expression for the intensity as a function of pixel position, under the moving edge. The integral is evaluated as three definite integrals, which are then summed.

- The integral of the region from $t = 0$ to t_1 is zero.
- The region from t_1 to t_2 is a straight line. By geometry this integral is $E_0.(t_2 - t_1)/2$.
- The integral of the region from t_2 to sT is $E_0.(sT - t_2)$.

Summed, these give an expression for the illumination at pixel $n + \Delta n$:

$$E(x_{n+\Delta n}) = \frac{E_0}{2}\left[2sT - (t_1 + t_2)\right] \tag{3.1}$$

t_1 and t_2 are

$$t_1 = \frac{x_{n+\Delta n} - x_n}{v} = \frac{\Delta n p_x}{v} \tag{3.2}$$

$$t_2 = \frac{x_{n+\Delta n} - x_n + p_x}{v} = \frac{(\Delta n + 1)p_x}{v} \tag{3.3}$$

which substituted in to Equation 3.1 gives a function which is a straight line with Δn.

$$E(\Delta n) = \frac{E_0}{2}\left[2sT - \left(\frac{2\Delta n p_x + p_x}{v}\right)\right] \tag{3.4}$$

This example supports the model of motion blur as a rectangular filter whose length corresponds to the length of the relative camera-object motion over the exposure time. It also enables a straightforward experimental design — the image of a moving edge is a straight line, which can be found by least-squares fitting to the appropriate samples. The slope of the straight line is proportional to the exposure time.

Since exposure time is set in the camera, it is possible to test whether motion blur behaves as predicted for a moving edge by comparing estimates of exposure time derived from motion blur length with camera exposure time. Section 3.4 describes an experiment to make this comparison.

3.3.4 Motion blur as filtering

Blur arising from front-on, planar motion can be modelled as a rectangular filter $k_{\mathbf{v}}(x)$ acting on a notional unblurred image $I_s(\mathbf{x})$:

$$k_{\mathbf{v}}(x) = \begin{cases} 1/L & \text{if } -L/2 < x < L/2 \\ 0 & \text{otherwise.} \end{cases}$$

where L is the length of the motion in pixels. The 2D filter is created by taking a rectangular region of width one and length L and rotating it so the long edge is parallel to $\mathbf{v}$. In the sampled image there will be some errors arising from quantisation, but these will be small. The image containing motion blur is then computed as

$$I(\mathbf{x}) = k_{\mathbf{v},\mathbf{x}}(\mathbf{x}) * I_s(\mathbf{x}) + n(\mathbf{x})$$

where $*$ is the convolution operation and $n(\mathbf{x})$ is camera thermal noise.

Because the 2D case can be decomposed into a rotation and a 1D filtering operation, evidence supporting the 1D model is sufficient to support the 2D model.

Applying a rectangular filter to a step function results in a signal with the same form as that derived in the previous section. Therefore, evidence gathered in an experiment measuring the blur caused by a moving edge will verify the rectangular filter model of motion blur.

3.4 Method

This Section describes the experimental method used to measure motion blur. Motion blur is measured by carefully setting up a camera and moving object to create a moving edge on the camera sensor, with known parameters. The shape of blur created by the moving edge is verified by calculating an estimate of the exposure time from the gradient of the image of the moving edge. The first part of this Section deals with the physical configuration of objects used to create a reproducible, constant motion. The second part deals with the software analysis carried out to produce measurements of motion blur from the captured video frames.

3.4.1 Requirements

The experiment had to meet the following requirements:

- A test signal which can be accurately modelled and precisely re-created.
- Use a simple signal amenable to analysis under experimental conditions which correspond to real broadcast or film use.
- Examine the behaviour of motion blur as a dependent variable on the following independent variables:
  - Exposure time (arising from varying shutter and framerate).
  - Motion length, arising from changing resolution.
  - Sensor type.

3.4.2 Physical configuration

The basic configuration was a camera pointing at a Technics 1200 direct drive record player ("the turntable".) A record player was chosen as it produces reliable motion whose speed is well-defined. Further, the tangential speed at any point in the image of a turntable is straightforward to compute, as the speed is directly proportional to the radius.

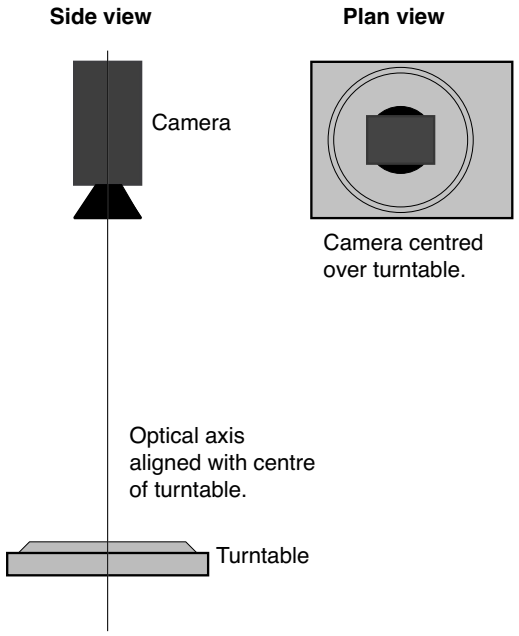


Figure 3.4: Ideal camera configuration: The optical axis is aligned with the turntable rotation.

The ideal configuration of camera and turntable has the optical axis of the camera passing through the centre of the turntable, and the rotating platter perpendicular to this axis. This configuration produces video frames containing circular images of the turntable platter. Samples can then be extracted on a circle centred on the turntable centre. If desired, these can be sub-sampled and interpolated to yield samples evenly spaced in angle, enabling straightforward analysis as a signal. Because the optical centre of the lens passes through the centre of the turntable, these extracted circles would each have uniform radial lens distortion. Therefore, no correction for lens distortion would be required. Figure 3.4 illustrates this ideal set up.

A "Rostrum" camera mount would have been ideal for this, but one was not always avail-

able. The following more general approach was used: The camera was mounted on a tripod which could point the camera nearly straight downwards, and the turntable placed on the floor beneath it with each corner supported independently. Several separate stages of adjustment were made to achieve the best possible alignment:

1. Initial estimate: Position and orientation of the turntable adjusted until the spindle is close to the image centre, and orientation very roughly correct.

2. Align optical axis and turntable spindle: The optical axis is estimated by observing the motion of the turntable centre as the focal length is varied. The turntable position is adjusted to align the turntable centre and optical axis.

3. Align turntable platter with optical axis: A mirror is placed centrally on the turntable. The position of the camera, and orientation of the turntable, are adjusted so that the image of itself the camera sees is centred on the spindle.

The last two stages were usually carried out iteratively; it was difficult to adjust one without spoiling the other. In practice, the errors gradually converged until the errors were too small to distinguish by eye.

The magnitudes of the errors involved can be estimated: The cameras all had user aids which would mark the centre of the screen. This provided a reference point against which to judge the motion of the turntable spindle as focal length was varied. Although no formal measurement of the error in the centre position was made, a reasonable estimate would be the position of the centre is correct to less than 40 pixels.

Judging the position of the image of the camera lens was more difficult. The cameras did not offer any on-screen devices against which the symmetry of the image of the camera could be measured. A reasonable assumption is that when the image of the camera is placed 10% of its width away from the centre, the error becomes noticeable. The turntable was typically placed just over a meter from the camera. Based on those distances and the diameter of the camera lens housing, if the reflected image of the camera was off by 10% of its width, then the turntable would be misaligned by less than one degree.

What would be the resultant deviation in circularity? In the affine approximation, the diameter of the turntable would be reduced to $cos(\theta)$ of the diameter when imaged perpendicularly. $cos(1) = 0.9998$, so the error in circularity will be small. The iterative approach taken to aligning the equipment meant that these errors are all as small as practically possible.

Figure 3.5 shows how the equipment was arranged for three of the four test shoots. The ARRI ALEXA S is supported by a dolly instead of a tripod, allowing it to be positioned with its optical axis almost vertical. The illumination for the RED, FOR.A and Sony PMW-500

was provided by a pair of "Kelvin Tiles", a brand of LED lighting. The illumination for the ARRI shoot was provided by a pair of incandescent lights, pointed at the ceiling. (The room used for the ARRI shoot was designed to be used in this way.)

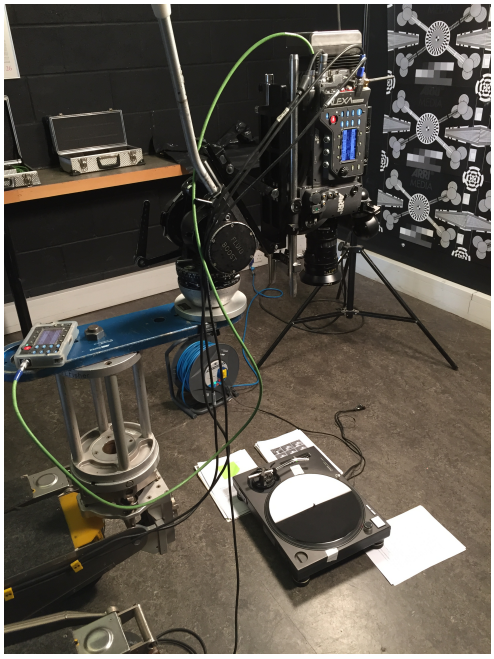
3.4.3 Subject

The high contrast edge was created with black felt and office paper. A circle of black felt the size of the turntable was cut, with a hole in the centre for the turntable spindle. A semicircular piece of white paper was cut, using the manufactured edge as the line across the centre of the circle. A small notch was made at the centre of the edge to accept the spindle.

The alignment of the white paper across the centre of the turntable was performed manually, before each capture. Before any experiments were carried out, two markers were fixed on the turntable, opposite one another, using the following method: The piece of paper was placed such that it lined up with these two markers. The turntable was then turned half a rotation and the alignment was checked. The positions of the paper edge, and markers, were changed until the alignment was subjectively correct with the edge in either position. Once this procedure has been finished, the markers on the turntable were made permanent. Before each experiment, the piece of paper was placed on the felt, lined up with the markers, and the centrality checked against the markers by eye, in both positions. Clearly this method will include some small error in the position of the edge. This was the most accurate method available.

(a) RED EPIC

(b) FOR.A FT-ONE



(c) ARRI ALEXA S

Figure 3.5: The arrangement of camera, turntable and lighting.

Procedure

Once all the equipment was aligned, videos were recorded with whichever independent parameters were available per camera. Those parameters were:

- Shutter (as a proportion of the frame duration)
- Frame duration
- Resolution
- Shutter type

Experiments were performed with four cameras; a Sony PMW-500, an ARRI ALEXA-S, a For-A FT-ONE and a RED EPIC. Their relevant characteristics are given below:

Sony PMW-500:

- 25fps at 1080×1920. 25fps and 50 fps and 720×1280
- 3-CCD sensors, with global electronic shutter.

RED EPIC:

- up to 200fps at 4096×2160 & 1920×1080
- Single CMOS sensor, with rolling electronic shutter.

ARRI ALEXA-S:

- up to 100fps at 1620×2880.
- Single CMOS sensor, with rolling electronic shutter[1].

FOR-A FT-ONE:

- up to 800fps at 4096×2160.
- Single CMOS sensor, rolling electronic shutter.

[1]This camera has a very fast rolling shutter readout, which emulates a global shutter very well.

3.4.4 Post-processing and data conditioning

In order to make useful measurements of motion blur length, some pre-processing is required. Each video sequence of the rotating edge is exported to single-frame images, then goes through the following pipeline:

- Read image.
- Undo gamma correction.
- Convert to luminance, or luminance proxy.

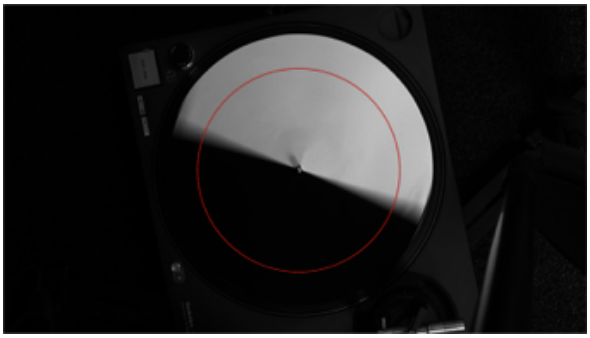
This results in a video frame in linear luminance. For some cameras tested it was not possible to find a reliable description of the colour space. In experiments examining the high-contrast edge, colour does not matter, so the green channel is used as a proxy for luminance if luminance cannot be found. The centre and radius of the turntable in the image are found (in practice, this requires many images, so is computed once per sequence).

Finally, for each sequence an approximation to the illumination is computed. The illumination was set up to be as uniform as possible with a simple lighting set-up (Typically two stage lights.) By measuring the illumination at each pixel, the variation in illumination can be corrected for, to give normalised data across each image. The illumination is found by finding the mean value of every pixel, but only when it is imaging the white half of the turntable. These are stored in an *illumination* file for each sequence. The normalisation will only be correct up to the variation in illumination of the white paper, under an amount of motion blur which will vary with radius. The error resulting from this variation is expected to be very small, and never *appeared* to come close to the error resulting from thermal noise in the camera.

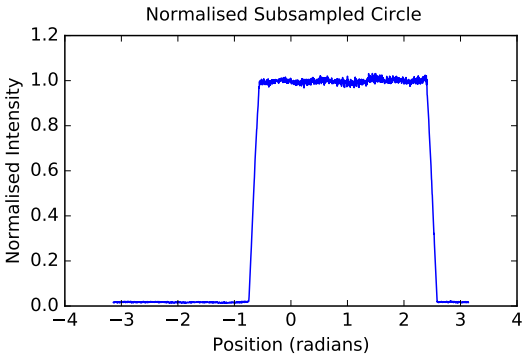
Circle Extraction and Analysis

Individual measurements of motion blur length are made by extracting circles of pixels from the luminance images. The data is noisy, and for some cameras corrupted by rolling shutter artefacts, so many circle measurements are needed. This section describes the process by which circles are extracted and analysed to give an estimate of motion blur length.

Circles of pixels are extracted using Bi-Quadratic interpolation of the luminance image at sites regularly spaced around the circle. They are normalised by dividing by an identically extracted set of pixels from the corresponding illumination file. This results in an array with the characteristic shape shown in Figure 3.6b. There is a “white” region with a value around 1, a “black” region with a value around 0, and two “edge blur” regions connecting them.



(a) The typical appearance of the turntable in a frame.



(b) The characteristic shape of a circle of pixels.

Figure 3.6: Circle extraction.

The motion blur length is computed by fitting a straight line to the edge blur regions and measuring the length in pixels between where this line intersects the white and black regions either side. Lines are fit to the white and black regions by either fitting a line to the samples, or assuming they have zero gradient, taking the average, and using the horizontal line at that average value.

Fitting a line to this data automatically is not a trivial task. Figure 3.7 shows an edge blur ramp in detail. Observe that there is a smooth curve between Positions 1.70 and 1.75, and between 1.85 and 1.90. This is not the modelled behaviour. The slight smoothing of the ideal sharp corners is the result of several very small point-spread functions in the optical path:

- Imperfection of the lens.
- Imperfection of the focus setting by the operator.
- Optical antialiasing low pass filter.

Four points must be chosen which divide the circle of samples into “white”, “black” and edge blur regions.

To initialize the process of finding the dividing points the circle is smoothed with a low-pass filter, then local extrema are sought in the gradient of the smoothed circle, with some local suppression of secondary peaks. These extrema are treated as estimates of the four transitions between sections of the circle. The result of this stage is verified by checking that the white and black regions are approximately the same length, and the two edge blur regions are approximately the same length. (In bulk processing, if this check is failed then this circle is discarded.) Next, a simulated annealing algorithm is used to refine the estimates of the transition points.

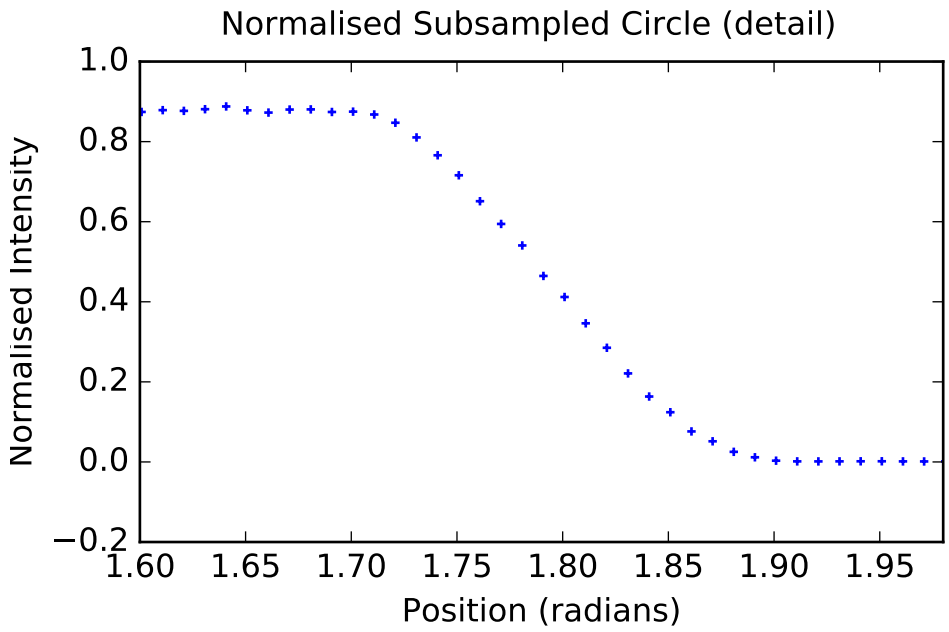


Figure 3.7: Detail of edge blur.

The cost function for simulated annealing is as follows: The circle is divided into four regions at the current four points. Straight lines are fitted to the samples in each of the four regions. Then, a weighted average of the perpendicular distances from the points to the line is taken, using up to 6 points from either end. (Fewer points are used only when the number of points in a region is less than 12.)

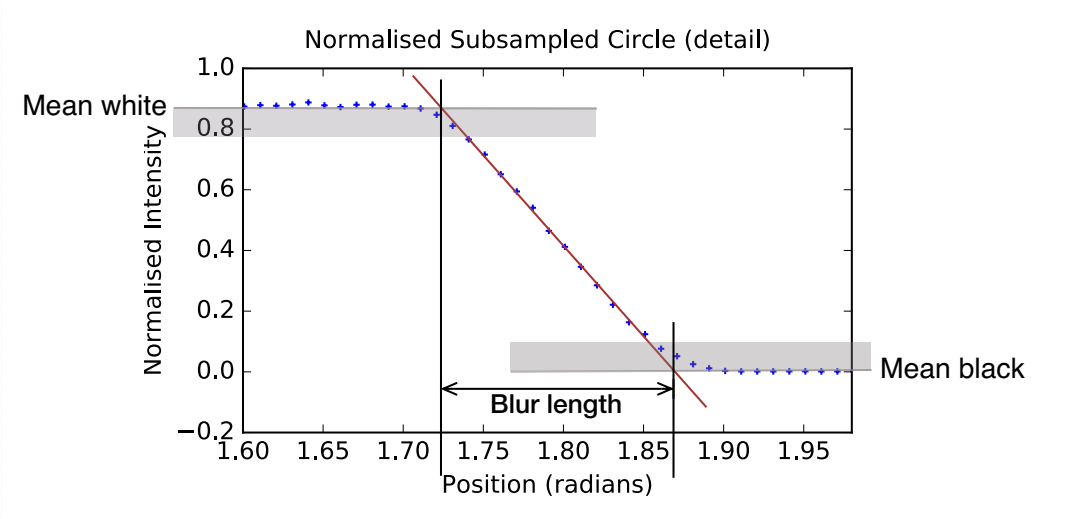


Figure 3.8: Estimating blur length by fitting a line to the blur ramp. The grey regions indicate excluded pixels distorted by the optical filters. The red line is the best fit to the remaining points.

The simulated annealing algorithm then proceeds as follows: If the cost of this set of transition points is lower than the last set, it becomes the new set. If it is higher, then it has a chance to become the new set with a probability which decreases with the number of iterations. Finally, one of the four intersection positions is selected at random, and randomly shifted up or down by one to give a new set of regions. The process is repeated until the cost function converges, or a maximum number of iterations is reached.

Once the final set of intersection points has been computed, the duration of the blur is calculated by fitting lines to the four sets of points. Optionally, some pixels in the edge blur region can be discarded in an attempt to mitigate the effect of the shape of the curve in the vicinity of the boundary between the edge blur region and the white and black regions. Figure 3.8 illustrates this.

Finally, the blur length in pixels is divided by the speed of the turntable at the appropriate radius in pixels per second. This gives an estimate of exposure duration which is independent of radius, which can then be compared between frames and sequences.

3.5 Results

Figures 3.9 to 3.14 present the measurements of exposure time based on motion blur length. Most plots are labelled "Expected exposure time" on the x-axis. This is calculated as Expected exposure time $= s/T$, where s is the shutter, expressed as a fraction with 1 meaning the sensor is exposed for the full duration of the frame. T is the frame rate in frames per second. (Both framerate and shutter are taken from the camera specification.) The y-axis shows the estimated exposure time derived from the experiments, as described above. The grey dashed lines in each of these plots shows $x = y$, which represents the value expected if the camera is behaving according to its specification, and the experiment is unbiased.

Figures 3.9, 3.10 and 3.11 show estimated exposure times where the camera exposure was varied using the shutter. In these figures each data point shows the mean estimated exposure time found from up to 10 circles extracted from 100 frames. (Sometimes circles would be automatically discarded.) The error bars indicate one standard deviation. In many cases the error is too small to be accurately represented on the plot.

Figure 3.9 shows measurements taken with the Sony PMW-500 at 25 frames per second. (a) and (b) were shot at 1920 × 1080 pixels, (c) and (d) at 1280 × 720 pixels. (a) and (c) were recorded using the standard gamma setting used for broadcast, corrected in the post processing. (b) and (d) were recorded in linear light. The short exposure experiments show a small bias to over-estimating the motion blur length, compared to the expected results based on the camera specification. Where the camera was set to a longer exposure, the estimated

exposure time tends to be lower than expected.

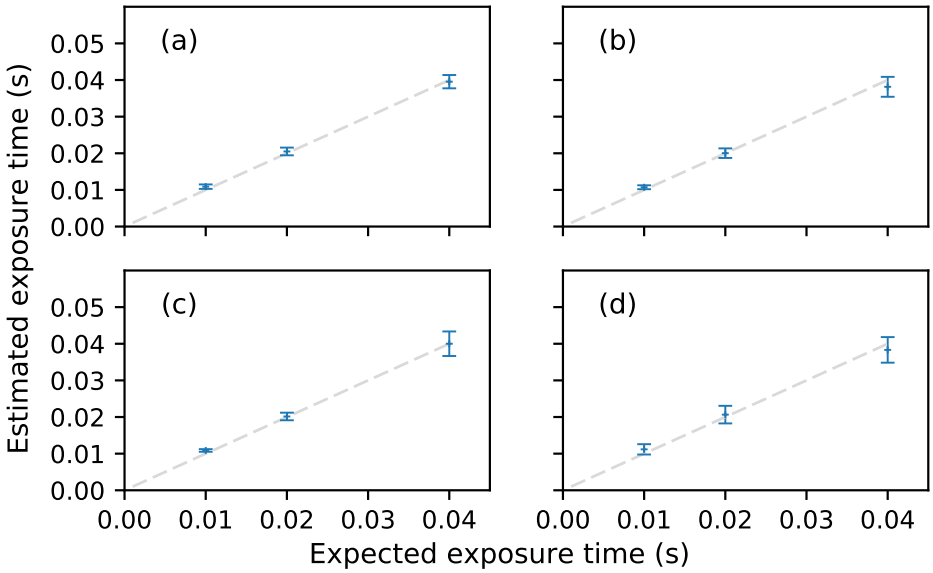


Figure 3.9: Motion blur length with varying shutter, Sony PMW-500. (a) 1080p25, Rec.709 gamma. (b) 1080p25, linear gamma. (c) 720p25, Rec.709 gamma. (d) 720p25, linear gamma.

Figures 3.10 and 3.11 give results from some other cameras where camera exposure time is varied using the shutter. Fig. 3.10 shows results where the camera was set to 50 frames per second, and Fig. 3.11 shows results for 25 frames per second. Each camera was using a different resolution. Estimates of exposure time based on the For.A camera are larger than expected from the specification, by 17-34%. The other cameras estimate the exposure time according to the manufacturers' specification, or over-estimate slightly.

Figures 3.12 and 3.13 show results from experiments where the shutter angle is fixed, and the frame rate varied. In Fig. 3.12 the shutter is fixed at 360 degrees, and in Fig. 3.13 the shutter is fixed at 180 degrees. Results from the ARRI ALEXA-S mechanical shutter mode are included here. There is no significant difference between the modes. As before the results from the For.A camera tend to over-estimate the exposure time compared to the camera specification. The other cameras usually estimate the exposure duration correctly to within one standard deviation.

Figure 3.14 shows 5 results extracted at different resolutions. Again, the measurements of exposure time are mostly in accordance with the manufacturers' specifications, with a small tendency to over-estimation. The measurements on the For.A FT-One once again over-estimate the exposure by more than the other manufacturers.

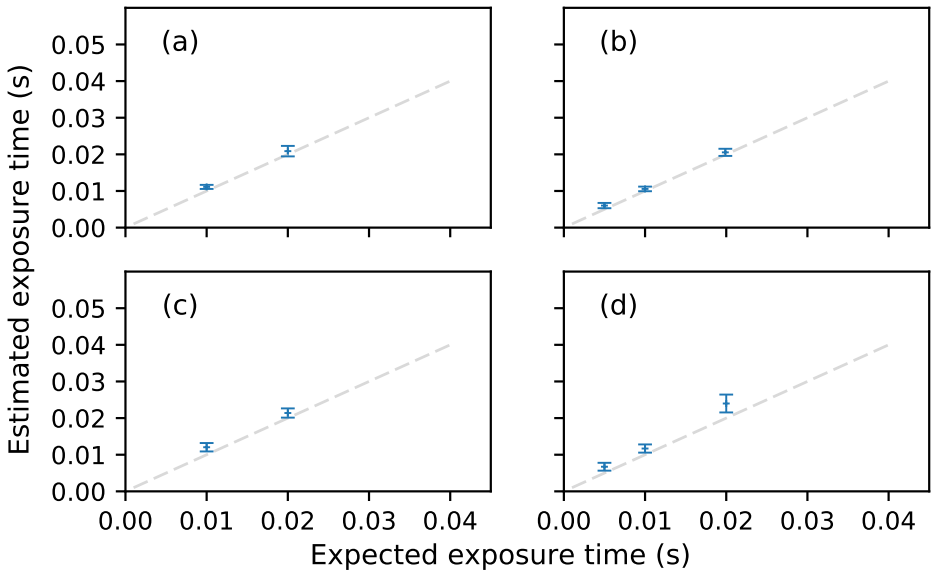


Figure 3.10: Motion blur with varying shutter, other cameras. (a) Sony PMW-500, 720p50. (b) ARRI ALEXA-S, 1620p50. (c) RED EPIC, 1080p50. (d) For.A FT-One, 2160p50.

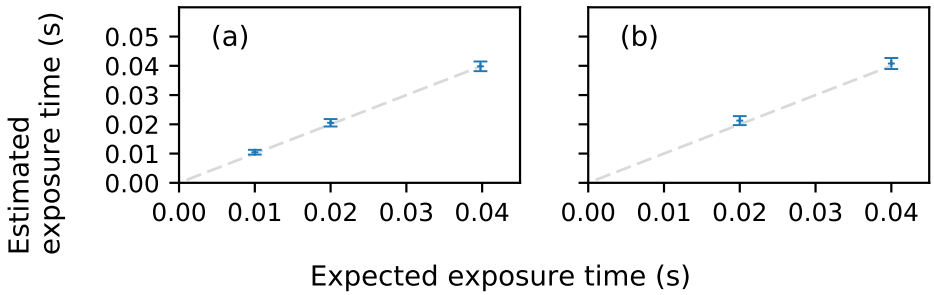


Figure 3.11: Motion blur with varying shutter, other cameras. (a) ARRI ALEXA-S 1620p25. (b) RED EPIC, 1080p25.

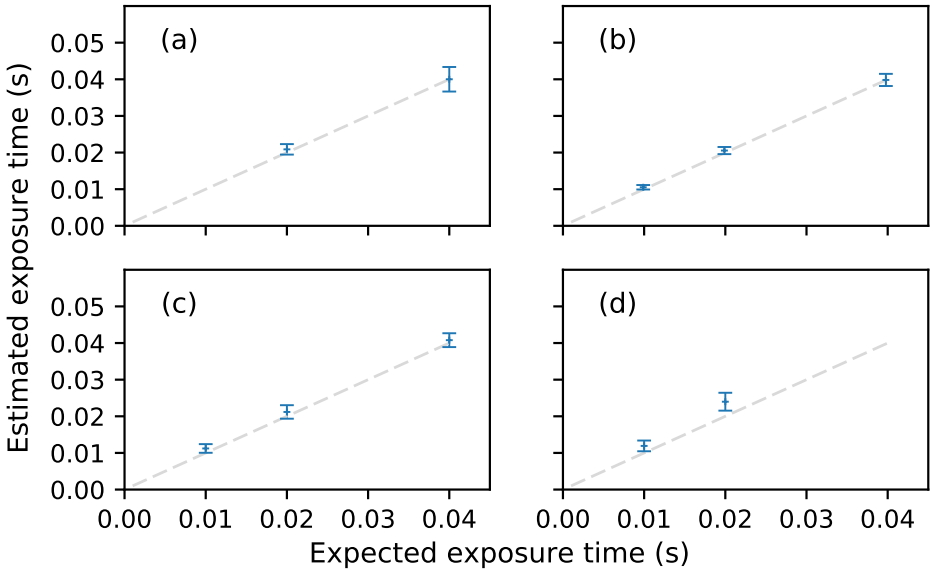


Figure 3.12: Motion blur with varying framerate, 360°shutter. (a) Sony PMW-500. (b) RED EPIC. (c) ARRI ALEXA-S. (d) FOR.A FT-One.

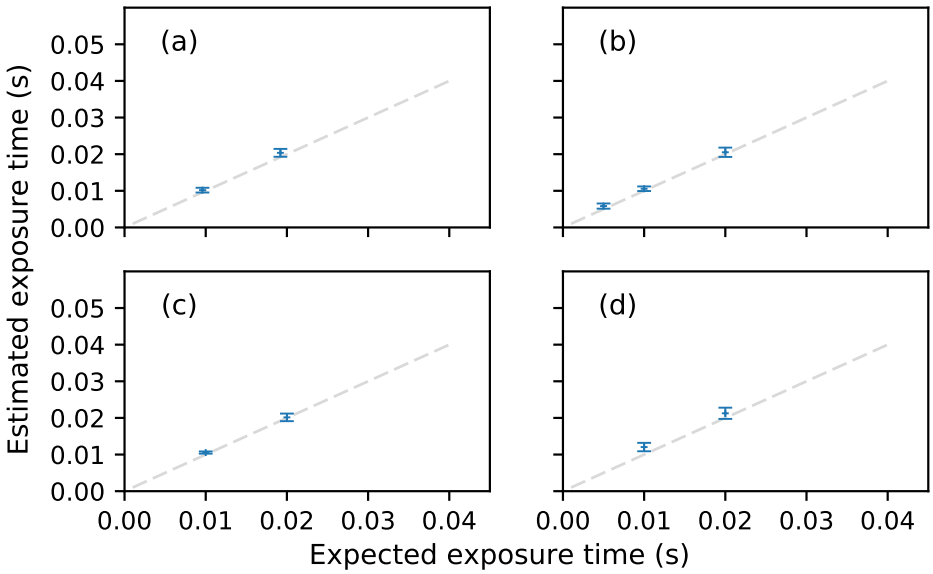


Figure 3.13: Motion blur with varying framerate, 180°shutter. (a)ARRI ALEXA-S mechanical shutter. (b) ARRI ALEXA-S electronic shutter. (c) Sony PMW-500 (d) RED EPIC.

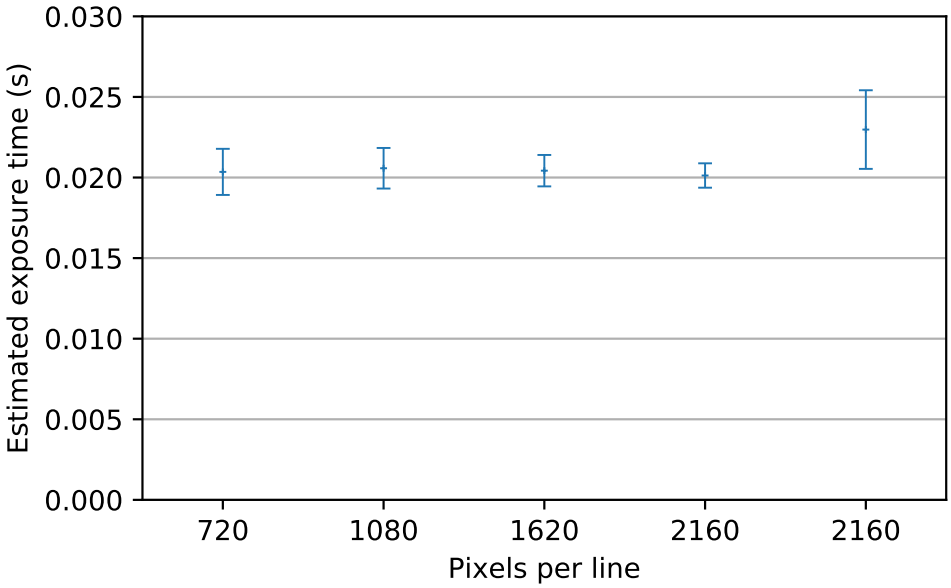


Figure 3.14: Motion blur with resolution. Exposure from camera settings is always 0.02s. Camera sources are Sony, Sony, ARRI, RED, For.A respectively.

3.6 Analysis

Overall the results of these experiments support the model of motion blur described in Section 3.3 up to experimental error. Two consistent deviations between the camera specified exposure time and the measured exposure time were observed:

- Estimates of exposure time tended to be longer than specified for short blur lengths and shorter than specified for long blur lengths.
- Measurements made using the For.A FT-ONE camera consistently over-estimated the exposure time compared to the specification by more than the other cameras.

Figures 3.15 and 3.16 shows these deviations explicitly. The difference between the camera specification of the exposure time and the measured exposure time is on the x axis, in units of standard deviations. The estimated exposure time is on the y axis. There is a clear trend to increased error with shorter motion blurs. The For.A camera always over-estimates by more than one standard deviation.

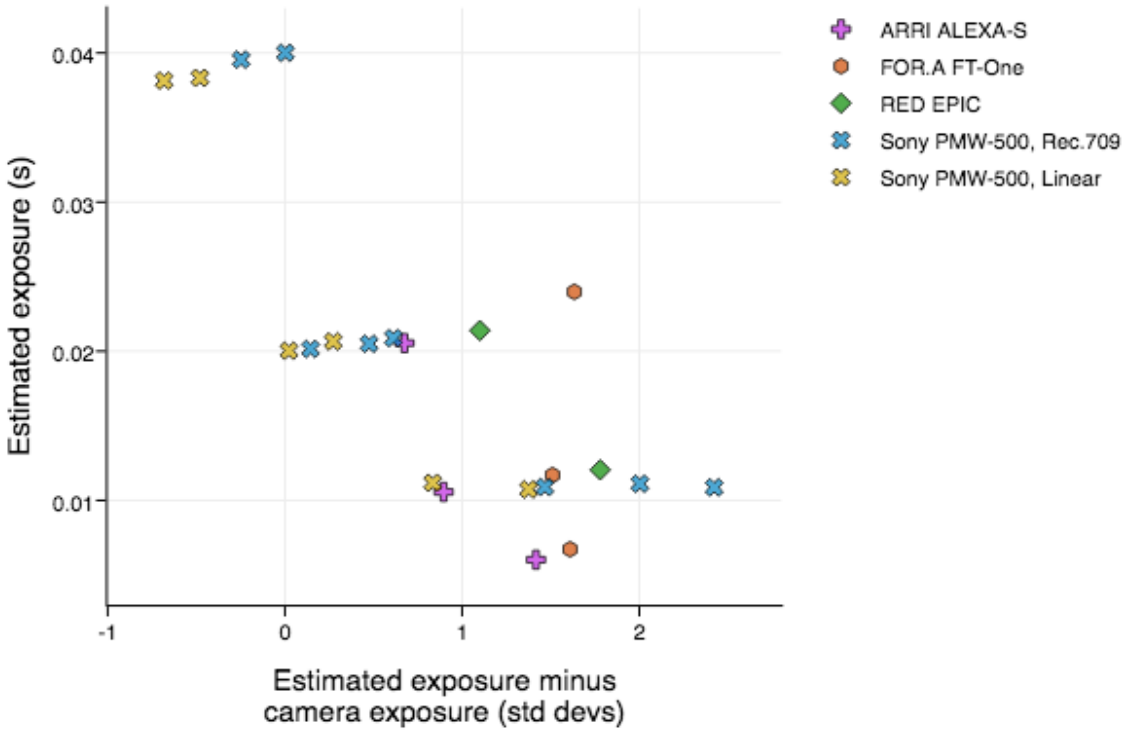


Figure 3.15: Difference between estimated exposure and specified exposure. Exposure varied using shutter control. Rec.709 and Linear refer to the gamma at capture.

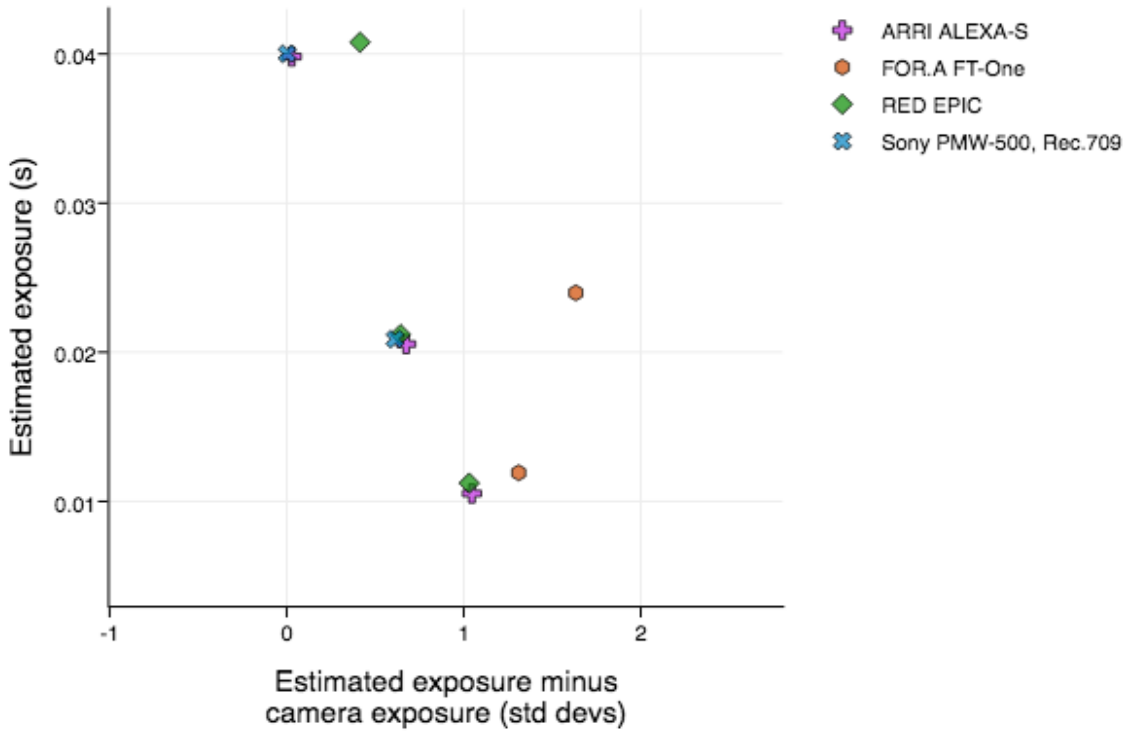


Figure 3.16: Difference between estimated exposure and specified exposure. Exposure varied using framerate control.

3.6.1 Error bias with exposure duration

In almost all measurements of exposures of 0.01s, and all exposures of 0.005s, the estimated exposure time is greater than expected by more than one standard deviation. The point spread function of the optical part of the camera has already been identified as a source of error. An attempt was made to correct for this by omitting some of the points near the transition between the edge ramp and white/black (see Section 3.4.4). This was done by specifying minimum and maximum values of normalised intensity to fit a line to. For a short blur, fewer samples are omitted, allowing more of the point spread function to contaminate the results. The method could have been modified to account for this, but a trade off must always be made between omitting samples from the edge blur region, and leaving sufficient samples in the blur ramp to fit a line whilst still suppressing noise.

3.6.2 For.A FT-ONE Results

Figures 3.10 and 3.12 show consistent over-estimation of exposure time in the results from the For.A. The alternative view in Figures 3.15 and 3.16 shows results from the For.A camera to

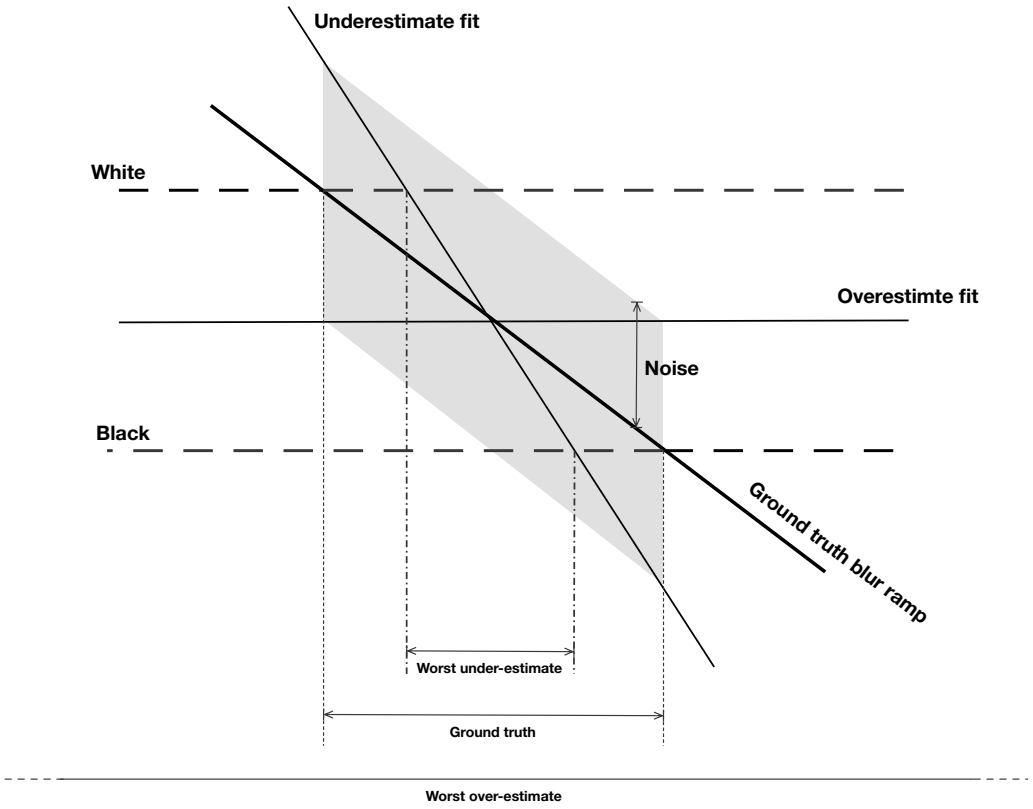


Figure 3.17: The tendency to over-estimate blur rather than under-estimate in the presence of noise.

have consistent bias with respect to the camera specifications, which the other cameras do not have. Note that the standard deviation is usually higher for the For.A compared to the other cameras, for a particular set of parameters. This means Figures 3.15 and 3.16 show the For.A results in a relatively favourable light. Yet still the difference between the experimental estimate of exposure and the camera setting is the greatest with the For.A.

The pictures from the For.A were subjectively more noisy than those from the other cameras. Excess noise can explain the higher standard deviation, and also explain the bias to over-estimate, in the following way: The method used to estimate motion blur is biased towards higher values in the presence of noise, as illustrated in Figure 3.17. The grey area indicates the possible bounds for a certain amount of noise to corrupt the signal by. The *Underestimate fit* line indicates the shortest possible under-estimate given this amount of noise, and the corresponding under estimate of the exposure time. The *Overestimate fit* line indicates the longest possible over-estimate given the same amount of noise. The corresponding estimate of the exposure time goes to infinity. From this example it should be clear that a sloping line

fitted in the presence of noise is biased to over-estimation of a quantity based on the horizontal projection of the line.

The pictures from the ARRI Alexa were less noisy than the other cameras - this camera contains a correction process for fixed pattern noise which is manually calibrated per-camera during the manufacturing process. These results better estimate exposure time, and have smaller standard deviations, which also supports the argument that excessive noise causes the error in the For.A FT-One camera results.

Figures 3.9, 3.10 and 3.10 show results matching the predictions of the model, when exposure is varied using the shutter. Figures 3.12 (a-c) and 3.13 show results matching the predictions of the model. Figure 3.12(d) overestimates blur length compared to the model, with deviation explained by the poor noise performance of the For.A camera, as discussed above. Figure 3.14 shows an estimate of motion blur at various resolutions. The noise performance of the For.A camera explains the over-estimate in the right-most data point. All other measurements show no discernible deviation from the model of motion blur, or deviation of the performance of the cameras from their settings.

3.7 Conclusion

The results of these experiments broadly support the model of motion blur as a rectangular filter for fronto-planar motion. Measuring short blurs is very difficult. Many of the results for very short exposures had to be discarded because they were too distorted by the optical point spread function. When trying to make sufficiently precise measurements using a video camera, noise is a significant source of error.

It has not been possible to determine whether any of the cameras mis-behave in any of the ways described in the Introduction to this Chapter. The combination of noise and optical point spread functions mean that the overall accuracy is low. To put the results in context, the standard deviations can be expressed in shutter angles. In Figure 3.9 the errors vary between 12 degrees and 125 degrees. In Figures 3.10 and 3.11 the errors vary between 9 and 43 degrees.

It is perhaps not surprising that the manufacturers of cameras and lenses which are considered by most to be of very high quality (and are amongst the most expensive available) should meet the idealised models. It would be a sensible design goal of a high quality video camera to make the effect of the various non-linear complexities involved in the whole processing chain to be indistinguishable from the unavoidable noise, and a very small blur in the optical path.

The experimental design is susceptible to noise and low spectral occupancy, when attempting to measure motion blur. But, were one to assume motion blur did follow the predictions of the models, then this experimental design could be useful as a measurement of image noise and

spectral occupancy. If a camera is being used for Television, Film, or other media where the end consumer is a human, it is desirable to have low noise and high spectral occupancy. Normally these properties must be determined with experiments requiring specialised equipment and calibration [83]. But this experiment requires only everyday equipment, and some studio lights. With some additional work to provide a calibrated scale of results, this experimental design could be useful for low-budget video makers who wish to find a high-quality camera without professional facilities.

The industrial contribution from this work is the experimental design, which is a cheap, accessible way to determine the quality of a camera. The academic contribution from this work is the observation that point spread function prevents this experimental design from achieving high measurement precision. Precise measurements of motion blur, and possibly other precise measurements of the temporal characteristics of cameras, will require a more sophisticated test set up. If, for example, objects were to move much more quickly, the effect of temporal integration would become greater than the effect of the point spread function.

3.7.1 A note on rolling shutters

The Red EPIC and ARRI ALEXA-S cameras have CMOS sensors. These can create a rolling shutter effect as a result of the exposure period for the line at the top of the sensor taking place slightly before the period for the line at the bottom, and varying continuously in between. The time difference between the first and last lines is called the readout time.

The ARRI also has a mechanical shutter, which when engaged can mitigate the rolling shutter effect by physically blocking light to and from the sensor at both the start and end of the exposure. The effect is not removed entirely, as the mechanical shutter also moves across the sensor in some non-zero time. However, it is not on the focal plane, and as such the image of the shutter edge will not be in focus on the sensor. (It was not possible to capture an image of the shutter edge directly using the camera.)

The Sony PMW-500 has a CCD sensor. The shutter behaviour is global, in that the exposure for each pixel begins and ends at the same time. For.A do not provide any information about their sensors.

A pre-experiment was carried out with a lower-quality camera. A frame was taken showing the turntable with the high contrast edge running vertically through the frame. By fitting a curve to the image of the edge, it was possible to measure the readout time of this camera. The same procedure was tried with the ARRI ALEXA-S, in electronic shutter mode, and the readout time was sufficiently small that it was masked by the (very small) point spread function and image noise.

Rolling shutter was taken into account in this experiment in the following way: When the rotating edge was near horizontal in the frame, the blur ramp would appear shorter on one side of the frame, and longer on the other side. Because all points on the turntable are the same distance from the camera, the effect is symmetrical. The errors cancel out by including two measurements of blur length from each sampled circle.

During this experiment it was observed that the ability to estimate the exposure time accurately was sensitive to image noise and the point spread function. Low noise and image sharpness are desirable qualities in many imaging applications, which are difficult to quantify. This experiment could be used, without modification, to measure the combined noise and point spread function of a camera-lens system. Whilst it is not possible to measure either quantity directly, the measurement of exposure time will only be measured precisely if the image noise is low, and the spectral occupancy is high. The results in this Thesis could be used as a baseline for comparison. In the future subjective tests could be used in conjunction with this experiment to develop a set of thresholds of image quality. The equipment needed to reproduce this experiment are easy to acquire, which is a benefit for anyone with limited time or budget.

Chapter 4

Assessing Feature Matching — Experimental Method

4.1 Introduction

In the following two Chapters, methods for dealing with motion blur in template-based matching and feature descriptor-based matching are proposed and evaluated. The methods proposed, and the experimental results describing their performance, constitute answers to question 2 posed at the beginning of this Thesis: "Can these models improve visual tracking in situations of differing motion blur, in real time?" This Chapter describes the experimental method used for the evaluation. The experimental design has two goals:

1. To produce results suitable for comparison with those in [1].

2. To measure suitability of a method to match sharp images to ones which might be corrupted by motion blur.

The experiment is based on the data published by Gauglitz et al in [1]. By following their method as far as possible, the results in this Thesis are directly comparable with [1].

Sections 1.1.1 and 1.1.2 motivated the problem of matching image features containing motion blur to those without blur. In order to investigate this problem in a formal way, a set of data of sharp and motion blurred images is required. The data provided by Gauglitz et al [1] includes a "reference" tracking mode, which provides this data exactly. It includes ground truth geometry between all camera frames to enable all inter frame motions to be computed. This data also allows for performance with varying motion blur to be varied.

This Chapter is structured as follows: Section 4.2 describes the data set provided by [1] in detail. Section 4.3 describes the experimental procedure. Section 4.4 describes the method for

evaluating performance, and Section 4.5 concludes the Chapter.

4.2 Data Set

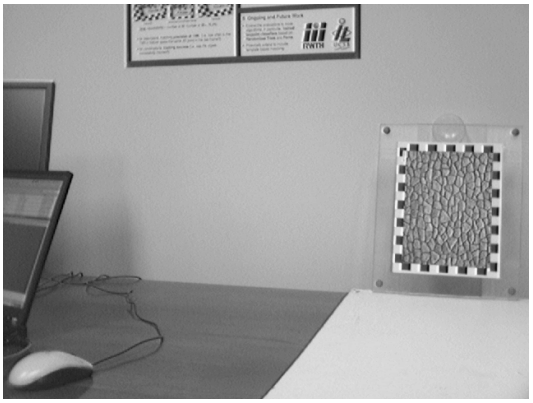
The data set published by Gauglitz et al [1] includes six sets of nine sequences of a camera panning past a planar target. There are nine different panning speeds of approximately integer multiples of 5.1 pixels per frame, and six different targets. Ground truth homographies relating the position of the planar target in each frame are provided. The data set included lens calibration information, calculated using OpenCV [86].

The camera is stationary in the first frame of each sequence, and is accelerated up to speed over a small number of frames after which the speed remains constant. The targets are shown in Figure 4.1.

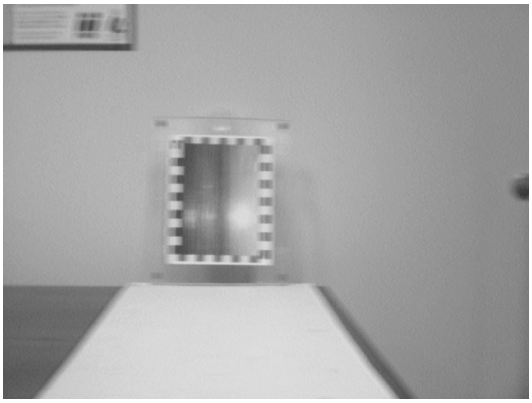
Figure 4.1: The targets included in the data set from Gauglitz et al. From left to right Wood, Bricks, Building, Paris, Mission, Sunset

Some care was taken by Gauglitz et al in the computation of the ground truth homographies. A mount was made to place the tracking targets within. The mount included four red balls, and had been engineered such that the centres of the balls were coplanar with the planar tracking target. A canonical reference frame was defined in which the balls form a rectangle. The position of all four balls was determined using an offline multi-step template tracking algorithm. The relative positions of the balls between pairs of frames was used to estimate a homography. Each video frame was warped to the canonical frame, and a user-guided image alignment process was used to finalise the homographies.

Figure 4.2 shows two example frames from the UCSB data set. On the left, the first frame from the Bricks sequence at speed 1 is shown. The target is in the mounting frame, with the coplanar balls visible at the corners. The target has not yet begun to move. This frame is the source of sharp features. On the right, a frame from the middle of the sequence showing Sunset at speed 6. In all sequences, only the target is used for the tracking. The chequerboard pattern around the perimeter is excluded. This is a small region, approximately 120×90 pixels. The software ignores the rest of the image.



(a) The start position used for the motion blur sequences



(b) Example frame mid-motion. Speed 6 of 9.

Figure 4.2: Example frames from the UCSB dataset.

A small webcam style camera was used to capture these sequences. To analyse how representative this data is of broadcast camera footage, the spectral content will be examined. The spectral occupancy plot in Figure 4.3 shows the mean horizontal log-spectrum of some test images. The horizontal spectrum is chosen as it allows direct comparison between fields from broadcast cameras running in interlaced mode, with a picture from a still camera, and the data in this data set, which is captured in progressive mode.

Figure 4.3 compares the spectral occupancy of some frames from this data set with an identically sized crop from a broadcast camera and a high-quality stills camera. Log-spectrum is plotted against frequency. Frequency is in units of the sampling frequency.

Kiel Harbour is a BBC test image taken with an SLR camera, and scanned. It is a useful image, as it is a very sharp image containing lots of high frequency detail, as well as some planar areas. It has been in use for more than 30 years is broadcast research at the BBC. This image clearly contains a lot of information at all frequencies.

The line labeled “Broadcast Sand Pit” is the occupancy of a long jump sand pit as seen by a broadcast camera. The lower half of the Sand Pit spectrum has comparable energy to those from the UCSB camera, but the upper half contains more energy. Later, it is noted that features are detected in the second octave. based on this spectral occupancy plot, there is a broadly similar amount of energy in the UCSB data as in a typical broadcast camera picture. Finally, it is worth noting that most of the energy in the UCSB Sunset picture is in the horizontal (And therefore appears in this plot.)

When covering live events, and particularly sports, broadcast cameras often use filters which boost the high frequencies in images, in order to make edges appear sharper. Usually called “aperture correction”, the filters are often implemented to introduce minimal delay, and hence

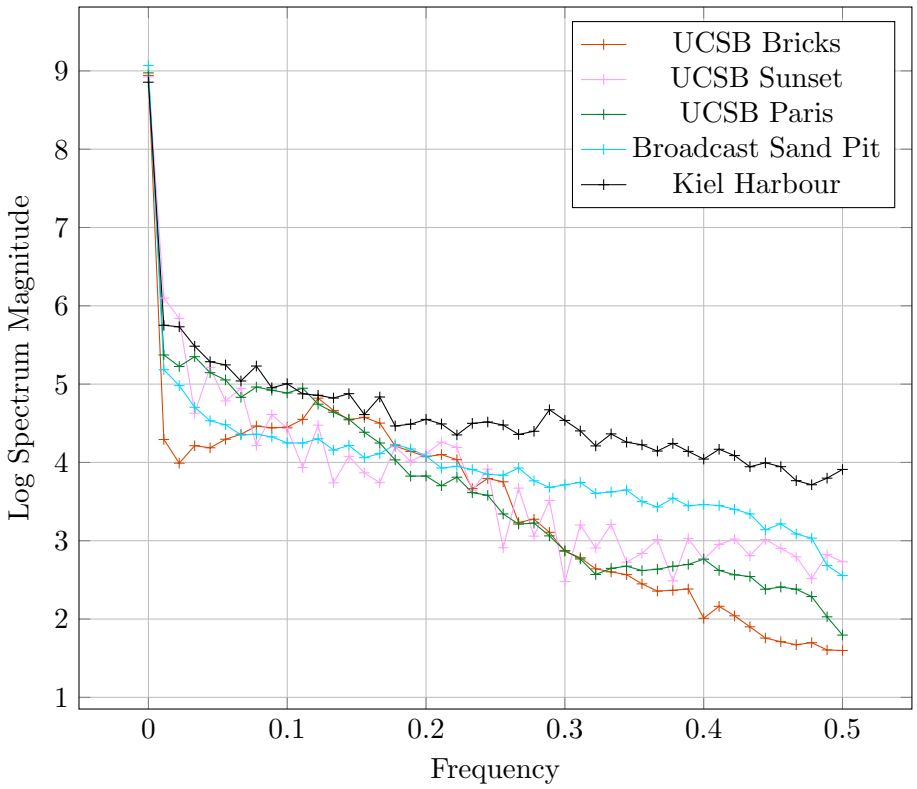


Figure 4.3: Spectral Occupancy of various signals.

are not symmetrical. The camera used in the UCSB data set has a similar filter. The effect can be seen by examining the sawtooth signal from the chequerboard pattern around the UCSB frames. The effect of this filter is shown in Figure 4.4. Although this makes the pictures look un-natural when viewed closely, this actually reflects quite well how real broadcast pictures look.

4.3 Procedure

The experimental procedure is described below.

Preparation, for each sequence, takes the following steps:

- Preprocessing: Remove lens distortion using supplied homographies, convert to luminance.
- (Phase Correlation only) remove gamma.
- Detect Harris corners in the first frame.
- Warp these corners to other frames in the sequence using supplied homographies.

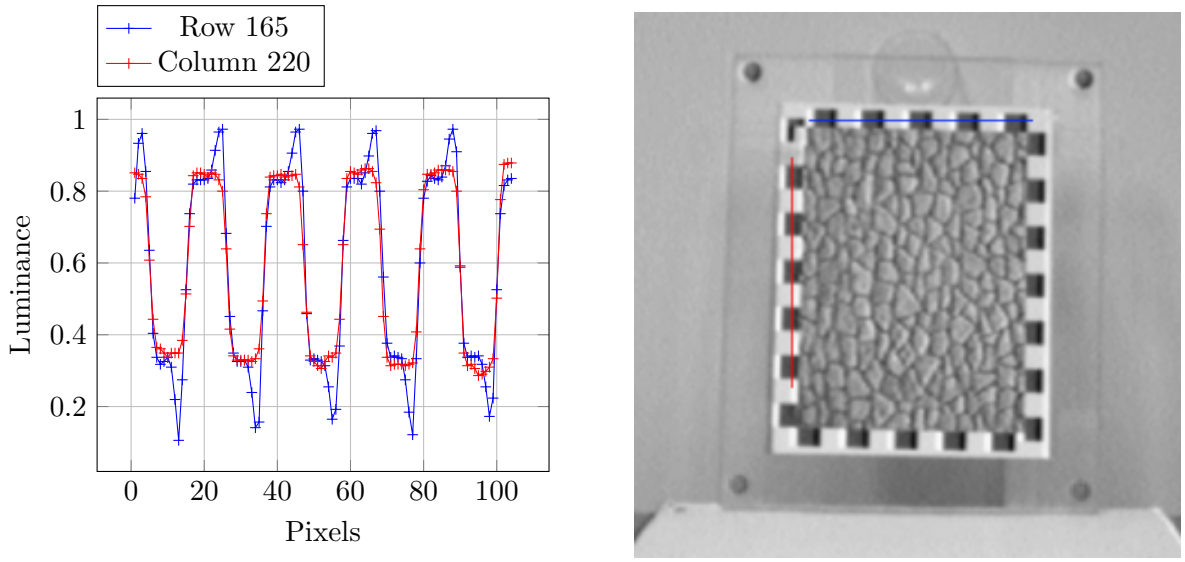


Figure 4.4: Left: A row and column of pixels showing the effect of the sharpening filter on horizontal edges. Right: Image crop showing the pixels extracted.

Then, two modes of tracking are tested: In both methods, the aim is to match features from a template frame to those in an input frame. In *Inter-frame* tracking the template and input are consecutive frames. *Reference* tracking uses the first frame in the sequence as the template frame, and searches for matches with each of the remaining frames in the sequence. Phase correlation-based methods (assessed in Chapter 5) and SIFT-based methods (assessed in Chapter 6) take different approaches to matching:

SIFT In Inter-frame tracking the co-ordinates of a feature in the template frame is used as the centre of a circular search region of radius 50 pixels in the input frame. The matching score is found between each feature in the template frame and the corresponding search region. In Reference tracking the co-ordinates of the feature point in the frame preceding the input frame are used to initialize the 50 pixel radius search region.

Phase correlation In both Inter-frame and Reference tracking, each feature in the template frame is matched against the corresponding feature in the input frame.

In order to isolate the impact of the proposed changes to tracking, features are not detected in every frame. Instead, the "perfect" feature detector approach from [1] is adopted. [1] showed that no feature detector achieves high repeatability under strong motion blur, even allowing for a 2 pixel reprojection error between frame pairs. Adopting the "perfect" detector means that matching procedures are given the best chance of success. Variations resulting from particular combinations of detector and descriptor are removed.

The "perfect" detector operates as follows: Image features are detected on a 2-times down-sampled version of the first frame in each sequence. The provided ground truth homographies are used to find corresponding co-ordinates in the later frames in the sequence.

Gauglitz et al do not state which detector they use for their descriptor results. Informal experiments carried out as part of this work showed that the Harris corner detector [13] yielded results most similar to those published. The properies of the Sunset target are described next. From Figure 4.1 it is clear that the Wood target contains a similar amount of spectral information as Sunset.

The Harris detector did not detect a sufficient number of features for the targets Wood and Sunset to produce meaningful results. It is worth noting that this data set is designed to test image feature detectors as well as descriptors. Therefore it is not surprising that some test data yielded fewer features.

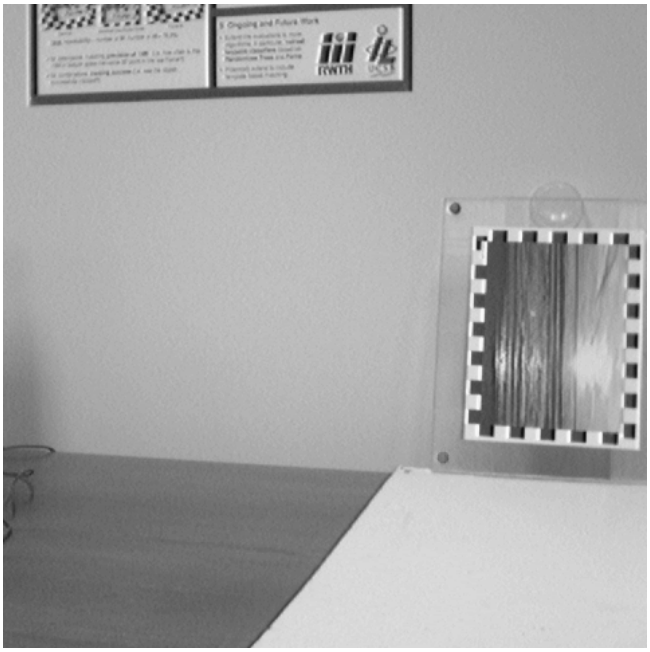
The Haar Wavelet decomposition can be used to visualise how energy in individual octave bands is distributed spatially within an image. It is a useful tool to understand the spectral content and signal to noise ratio of a camera when the camera is not available to run formal tests on. Figure 4.6 shows Haar wavelet decompositions of four image regions. The source image regions are shown in Figure 4.5. The colour images were converted to Luminance and all pixel values linearised and normalised before analysis. A gain of 5 and bias of 0.5 has been applied to make the low level high frequency signals more clearly visible. The wavelet decompositions have the horizontal energy in the top-right, the vertical energy in the bottom-left, and the diagonal energy in the bottom-right. The top-left quadrant contains the same decomposition for the next octave of the signal. The top-left quadrant which does not contain wavelet coefficients ordinarily contains the residual low-frequency energy. The processing to make the noise visible has put the low frequency energy out of the luminance range, so it is clamped to white.

The wavelet decompositions in Figure 4.6 (b) and (c) are fields extracted from an interlaced broadcast camera signal, so their vertical resolution is half the horizontal. This means that the vertical energy is not directly comparable with the vertical energy in the other decompositions. The top-left picture is a frame from the UCSB Sunset sequence, with no motion. The bottom-right is a crop from the BBC test image Kiel Harbour. The wavelet decomposition of Kiel harbour (Fig. 4.6 (d)) shows a relatively large amount of high frequency detail in all bands, and relatively low noise in planar areas. The two decompositions of broadcast camera data show significant high frequency detail, both on the athlete and the sandpit. Compare the amount of high frequency energy in the sandpit and the running track beyond to get an impression of the signal to noise ratio.

There are a few important differences in the UCSB Sunset wavelet decomposition (Fig. 4.6

(a)). Firstly, the amount of signal in the bottom-right part of the wavelet decomposition is very low, and comparable to the noise in the rest of this region. There is a lot of energy in the horizontal part of the decomposition but relatively little in the vertical. A similar pattern is visible in the second octave of the wavelet decomposition, although with a little more signal compared to noise. The Harris detector is tuned to find corners, where there is significant energy in at least two directions. Clearly the Sunset image has relatively little energy in the vertical, which is likely to be the reason the Harris detector did not find many features.

This data set is designed to test both detectors and descriptors. Including Sunset and Wood in the data set is a sensible design. It will help to identify detectors which are not able to detect features in image regions where there is only energy in one direction. Because the Wood and Sunset targets appear to have been designed to be difficult for feature detectors to operate on, it is prudent to remove them from this analysis.



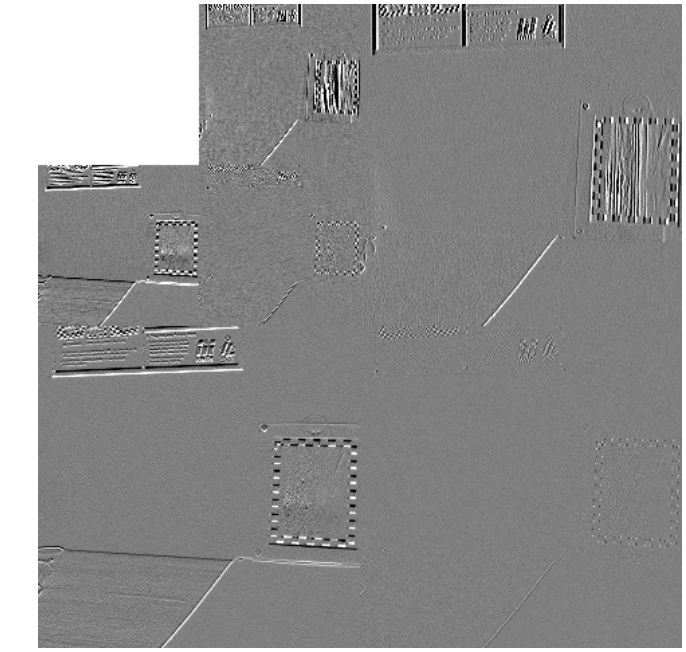
(a)

(b)

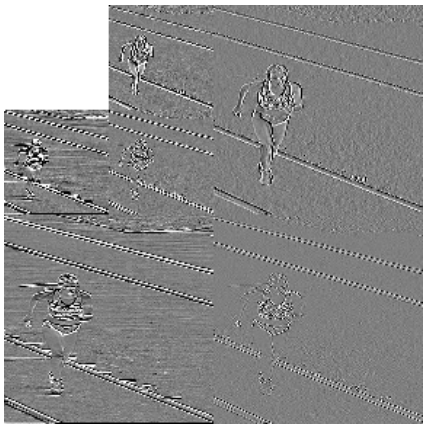
(c)

(d)

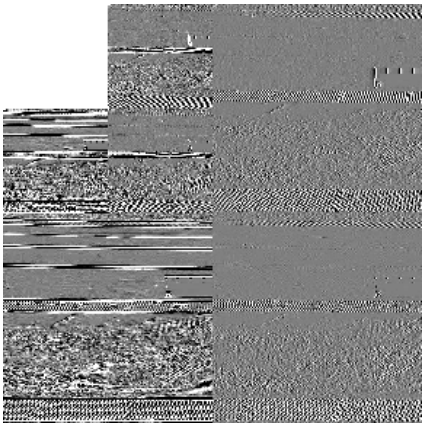
Figure 4.5: Example images for wavelet analysis. (a) UCSB Sunset; (b) Long Jump 1; (c) Long Jump 2; (d) Kiel Harbour.



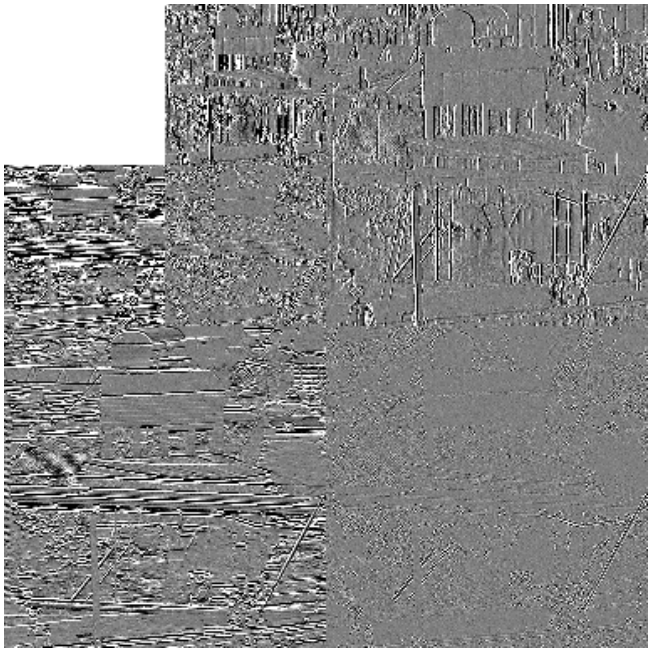
(a)



(b)



(c)



(d)

Figure 4.6: Wavelet decompositions. (a) UCSB Sunset; (b) Long Jump 1; (c) Long Jump 2; (d) Kiel Harbour.

4.4 Evaluation

Precision is used to measure the performance of the methods under test in Chapters 5 and 6. *Recall* (as paired with *precision* in [4]) is not used: It provides a measure of the discriminatory power of a descriptor over a large corpus of image feature points. This is not necessary in visual tracking because the camera motion is assumed to be small between frames, so the corpus of candidate features is constrained. *Recall* also has an implicit assumption that one image feature is being measured against many. This is not the case in phase correlation based tracking, so *recall* is not appropriate.

$$precision = \frac{number\ of\ correct\ matches}{number\ of\ correct + number\ of\ false\ matches}$$

Different methods are needed to distinguish *correct matches* from *incorrect matches* for descriptor-based tracking and template-based tracking. Phase Correlation, and the related methods examined in Chapter 5 are template-based. Each of these methods provides an updated location for a feature point in the input image, based on the template image and an initial guess. No measure of the quality of the match is produced, though. In these cases, a match is assumed to be correct if the algorithm results in a location within the image within some threshold distance of the ground truth position.

The SIFT-family methods examined in Chapter 6 do not update the position of the feature in the input frame. Instead they produce a matching score which describes the similarity of the template and input feature point. Matching scores are computed between a descriptor from the template frame and all the descriptors in the 50 pixel search window in the input frame. If the best matching score is found between the known-correct feature pair, and is less than 0.8 times the second best, then the feature point is assumed to have been correctly matched. (The 0.8 factor was originally proposed for this purpose by Lowe [20].) Otherwise the point is assumed to be a false match.

Following the method described in [1], Gauglitz et al do not assess feature matching methods capable of refining keypoint locations. However, when assessing integrated use of detectors and descriptors to track feature points, a threshold radius of 2 pixels is used to decide if a detected point matches a ground truth location. Based on this decision, a feature location found to within 2 pixels of the ground truth location will be taken to be a correct match.

4.5 Conclusion

This Chapter has described an experimental method which is designed to meet the aims given at the beginning. The experimental method mimics [1] as closely as possible. By measuring

precision with a suitable threshold, the suitability for practical applications in broadcast camera tracking can be assessed.

Different methods are needed for SIFT descriptor matching and phase correlation matching, as they operate differently and produce different outputs. The different methods are made closely comparable by defining similar *precision* metrics for both.

An analysis of the qualities of the data set is given. They are a compromise between testing feature detectors and testing matching techniques. In Section 7.1, suggestions are made for an improved data set.

Chapter 5

Velocity Corrected Phase Correlation

5.1 Introduction

This Chapter proposes a new modification to phase correlation, velocity corrected phase correlation. This approach is designed to improve real time visual tracking between images where one contains motion blur, as in question 2 at the beginning of this Thesis. Velocity corrected phase correlation is assessed using the experimental procedure described in Chapter 4, and is compared to the state of the art method [19].

As described in the literature review, phase correlation was chosen to be the basis for this method because it is fast enough for use in a real time tracking system, and there is a clear and simple method to undo the effects of motion blur on the matching process.

Velocity corrected phase correlation is shown to have the following advantages over the state of the art:

- Higher precision for longer motion blur lengths.
- Larger potential registration radius.
- Blur parameters are found as a side effect.

The computational requirements are shown to be equivalent, or negligibly more than the state of the art method.

This Chapter is structured as follows: Some background introducing Correlation-based methods is given in Section 5.2 from which velocity corrected phase correlation is derived in Section 5.3. An analysis of the computation time required by velocity corrected phase

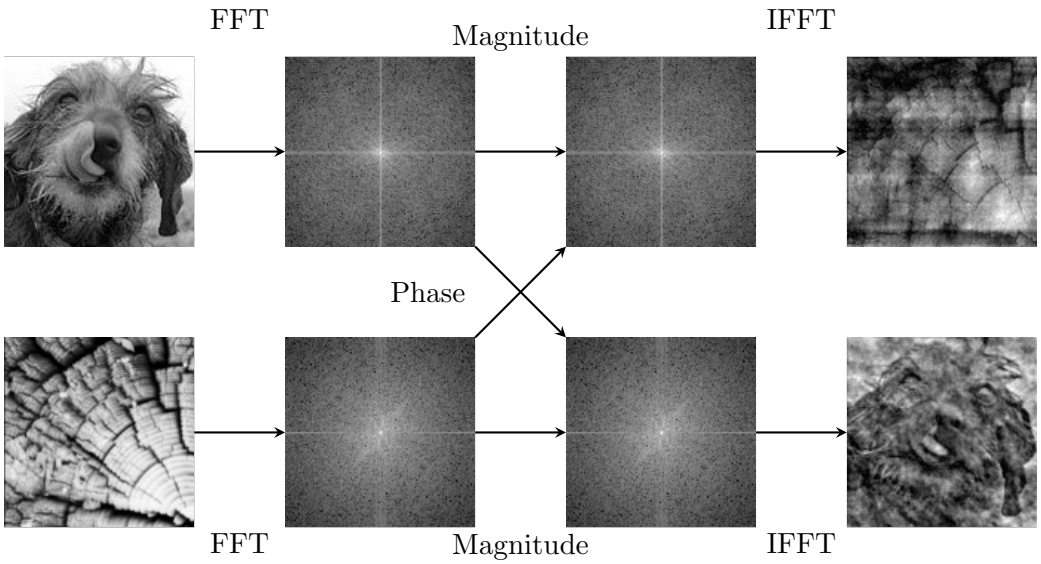


Figure 5.1: Phase carries the structural information about the content of an image. The dog's face is visible in the bottom-right image. The structure of the wood is visible in the upper right.

correlation, as well as the state of the art method, is given in Section 5.4. In Section 5.5 results are presented and discussed. The Chapter concludes in Section 5.6.

5.2 Background

5.2.1 Phase in Images

The phase information in the Fourier representation of an image represents more of the subjective appearance of an image than the magnitude. This can be shown with the simple experiment in Figure 5.1. The magnitude and phase information of a pair of images are swapped, in the Fourier domain. The dogs face is clearly visible in the bottom-right image. The wood structure is clearly visible in the top-right image. The phase information clearly carries the structural information about the image.

To find matching regions in image pairs, it is more important to consider the phase of the spectrum than the magnitude. In signal processing, the cross correlation function is used to identify relative shifts between two signals. The cross correlation function of a pair of signals f and g is given by

$$\hat{S}_{fg}(u) = \int \bar{f}(x)g(x-u)du$$

where $\bar{f}$ is the complex conjugate of f. The location of the peak of $\hat{S}_{fg}(u)$ indicates the relative offset at which f and g are most similar, ie $f(x-d) \approx g(x)$ if d is the location of the peak in

$\hat{S}_{fg}$.

This can be extended to two dimensions by replacing the signals on a single variable $f(x)$ with signals in two variables $f(\mathbf{x})$ where $\mathbf{x}$ is a 2-component vector (and similarly u becomes $\mathbf{u}$). Using the well-known relationship between the spatial domain product and Fourier domain convolution, the cross-correlation can be computed in the Fourier domain:

$$F = \mathcal{F}(f(\mathbf{x})) \ ; \ G = \mathcal{F}(g(\mathbf{x})),$$
$$\hat{S}_{fg}(\mathbf{u}) = \mathcal{F}^{-1}(\bar{F}G)$$

where $\mathcal{F}$ indicates the Fourier transform.

Phase correlation is closely related to cross correlation. During the computation of the cross correlation, the intermediate product $\bar{F}G$ is called the cross power spectrum. To find the phase correlation S_{fg} the cross power spectrum is normalised, by dividing each coefficient by its magnitude:

$$S_{fg} = \mathcal{F}^{-1}\frac{\bar{F}G}{|\bar{F}G|} \tag{5.1}$$

Because the phase correlation only responds to the phase, it is immune to linear changes in both brightness and contrast. (Assuming that the signal doesn't clip.) It is also immune to the magnitude of any filter applied to the image, and is only changed by the phase. This is the property exploited by veclocity corrected phase correlation. The immunity to Fourier magnitude is not total. Noise has an effect which must be dealt with, which is discussed in Section 5.3.6.

Figure 5.2 shows example correlation surfaces from cross correlation and phase correlation between two images. The top row shows the two input images. These are cropped from a larger image, at slightly different locations. A Gaussian blur has been applied to the upper right image. The lower left image shows the cross correlation result. The lower right image shows the phase correlation result. The green cross indicates the maximum. The red cross indicates the correct offset between images. The lower-right image has been magnified by a factor of 10 to show the detail in the peak. Even with the magification the phase correlation peak is much more sharp than the cross correlation peak. The phase correlation result is a much clearer, sharper peak than the cross correlation one. Phase correlation is also correct (The red and green crosses are on top of each other in the lower-right image.)

5.2.2 Sub-pixel location refinement

In an appendix to his Thesis [87], Thomas proposes a fast method for estimating sub-pixel peak location in S_{fg}, based on the assumption that the shape of S_{fg} is a sinc $(sin(x)/x)$ function.

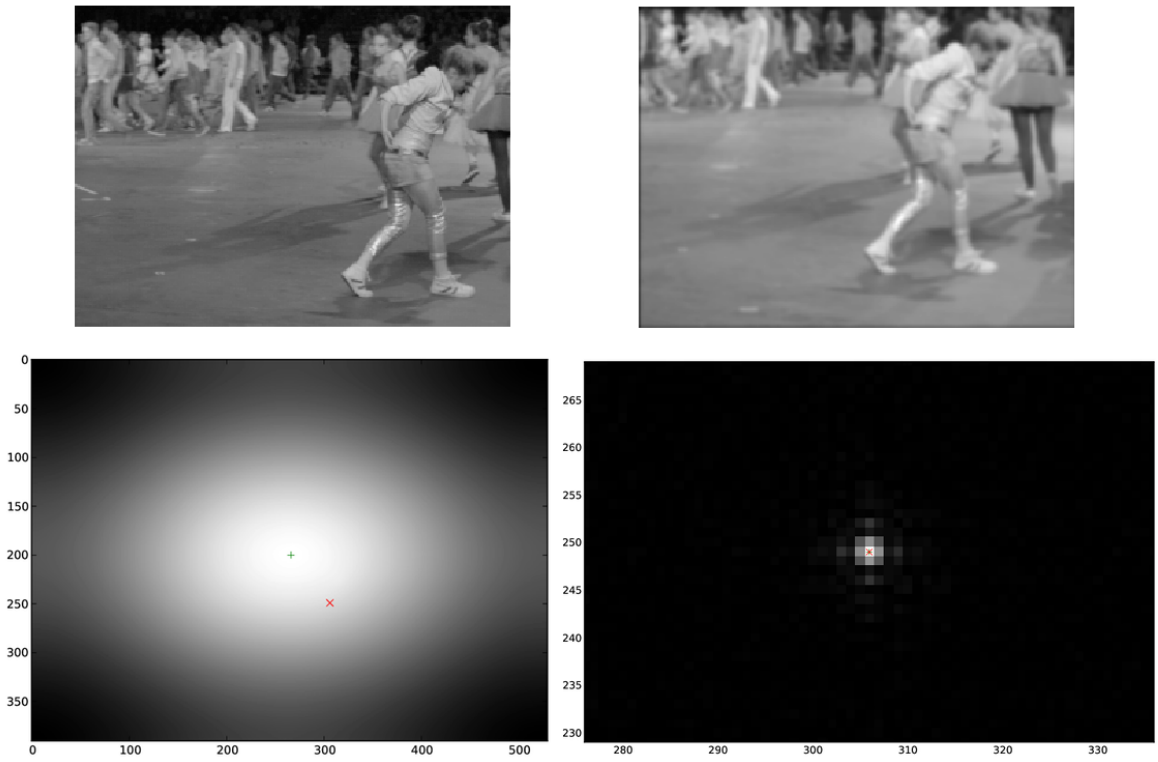


Figure 5.2: Comparing cross correlation and phase correlation.

If the value of S_{fg} at the peak is $S(d)$, the refinement method is to find

$$\delta = \frac{S(d+1) - S(d-1)}{2(aS(d) + bS(d+1) + cS(d-1))} \tag{5.2}$$

where

$$\begin{aligned} a &= 0.8, & b &= 0.2, & c &= -1, & &\text{if } S(d+1) > S(d-1) \\ a &= 0.8, & b &= -1, & c &= 0.2, & &\text{if } S(d+1) < S(d-1) \end{aligned}$$

The sub-pixel peak location is at $d+\delta$. (The derivation of the values of a, b, and c are given in [87].) According to the theoretical results, this approximation is correct to at worst 0.01 pixels. This method will be used to compute refined sub-pixel peak locations in these experiments.

5.3 Velocity Corrected Phase Correlation (VCPC)

5.3.1 The effect of motion blur on Fourier Phase.

As was verified by the measurements of Chapter 3, an image with motion blur $I(\mathbf{x})$ can be modelled using a rectangular filter $k(\mathbf{x})$ convolved with a notional unblurred image $I_s(\mathbf{x})$:

$$I(\mathbf{x}) = k_{\mathbf{v},\mathbf{x}}(\mathbf{x}) * I_s(\mathbf{x}) + n(\mathbf{x}) \tag{5.3}$$

When tracking a pan-tilt-zoom broadcast camera, it is usual to omit foreground information using a coarse process as described in Section 1.1.1. For a given frame, stationary background objects will suffer very similar amounts of motion blur. (This assumption will be violated if a fisheye lens, or very long telephoto lens, is used.) These experiments assume a normal lens will be used. It is also assumed that a camera is being tracked, and so a good first estimate of velocity is available.

The Fourier transform of the rectangular filter $k_{\mathbf{v}}(\mathbf{x})$ is a sinc function, which has positive and negative lobes. (See Figure 5.4.) Recall that, except for small magnitudes which bring the signal into the noise, phase correlation is immune to the magnitude of the filter, and is only effected by the phase. The filter coefficients corresponding to positive lobes will not modify the phase of the output signal, but those corresponding to negative lobes with change the phase by π. The result is that some components of the spectrum of $I(\mathbf{x})$ will be π out of phase with $I_s(\mathbf{x})$. This effect will be referred to as "phase inversion".

Recall the motivating problem is to match features from an unblurred reference image with those from a blurred input image. What impact does motion blur have on the phase correlation between sub-images where one contains motion blur? Ignoring noise for a moment, computing $\bar{F}G/|\bar{F}G|$ in Equation 5.1 is equivalent to finding the difference in phase between each coefficient in F and G. When the inverse Fourier transform is taken, the coefficients of $\bar{F}G/|\bar{F}G|$ interfere constructively to produce a peak at one location, and destructively everywhere else. If a motion blur filter has shifted the phases of some of the coefficients of one image by π, then this pattern of constructive and destructive interference will be broken.

The following experiment shows the effect; Apply an artificial motion blur to an image using a rectangular filter. Find the phase correlation S_{fg} between the image and the copy with artificial blur. Figure 5.3 shows a line of pixels from the result of this process. Two peaks are clearly visible, and neither is at the correct offset (0).

This shape in the correlation surface is characteristic of phase correlation in the presence of motion blur. Figure 5.6 shows that the inter-peak distance is equal to the blur length. A spatial domain interpretation of this observation is that the end of a blurred edge will match the unblurred edge better than anything else along the blurred image, so there will be peaks corresponding to a poor match at each end of a blurred object, but the match in the centre will be no better than any other part of the image.

It has been shown that motion blur interferes with phase correlation. The negative filter coefficients in the motion blur filter result in phase inversion. Consequently, S_{fg} no longer has a clear peak. A double peak structure is the characteristic result.

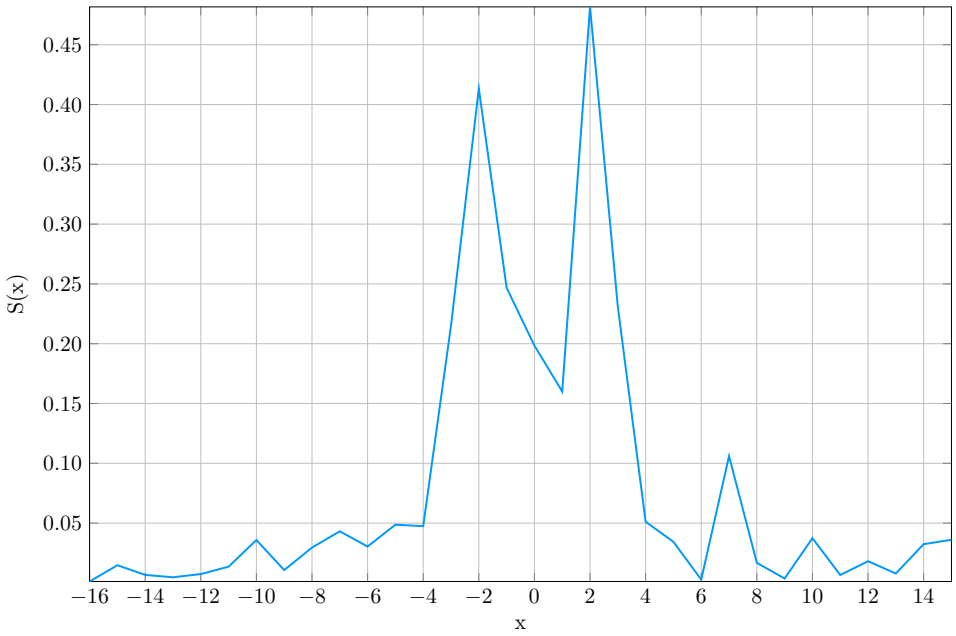


Figure 5.3: A line from the result of finding phase correlation between two copies of Kiel Harbour, where one has artificial motion blur.

5.3.2 Correcting for Motion Blur

The frequencies which have negative coefficients are predictable, given the velocity of the motion blur in the image. If the locations of these negative coefficients can be predicted, then they can be inverted again to restore the "original" phase correlation (not accounting for noise, and the attenuation into the noise floor of some signal coefficients by the filter.) The previous section showed how phase inversion arises from negative filter coefficients, in the filter model of motion blur. Here a correction mask is derived from the Fourier transform of the filter.

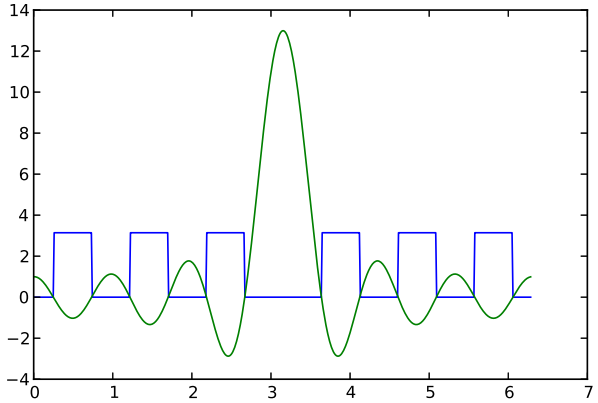


Figure 5.4: The real and phase components of sinc function.

Figure 5.4 shows the real and phase parts of a sinc function. The important characteristic to note is that when the real part is negative, the phase is π. Hence the mask required can be computed from the signs of the corresponding blur filter coefficients. The mask m is related to the filter k as:

$$K = \mathcal{F}(k) \tag{5.4}$$

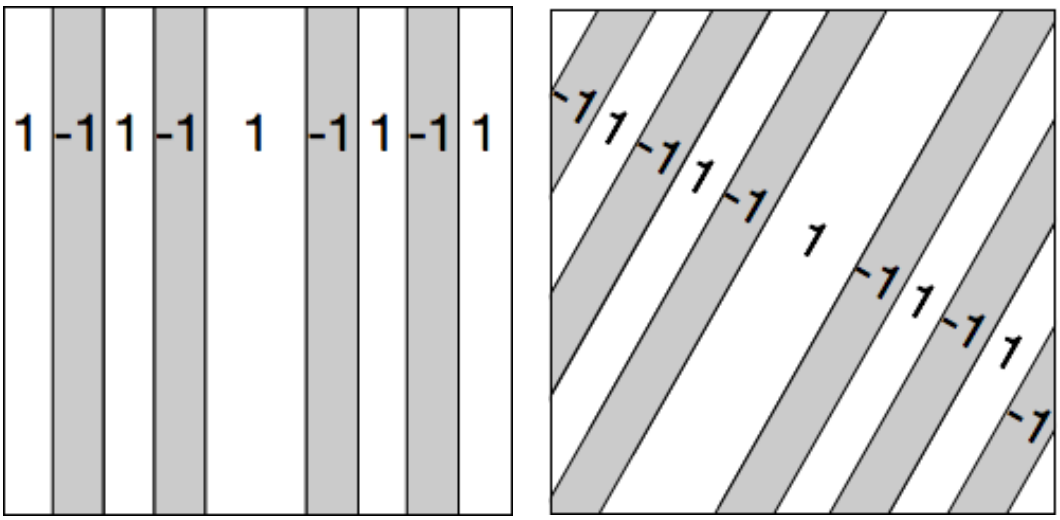
$$m = sign(K) \tag{5.5}$$

where $sign(x)$ is 1 if x is positive and -1 if x is negative. To correct phase correlation for motion blur, multiply the mask elementwise with the normalised cross power spectrum, and take the inverse Fourier transform:

$$S_{fg} = \mathcal{F}^{-1}\frac{m\bar{F}G}{|\bar{F}G|} \tag{5.6}$$

The mask introduces a further phase shift of π to each coefficient which has already been shifted by π, leaving other coefficients unaffected. After the application of the mask, each coefficient will have suffered a total phase shift of either 0 or 2π. This intermediate stage is equivalent to a corresponding stage between two images with no motion blur.

Figure 5.5(a) shows an example mask computed for a horizontal motion blur. Figure 5.5(b) shows an example mask for a motion blur at an angle to the horizontal. The masks are related only by a rotation.



(a) Horizontal blur vector. (b) Angled blur vector.

Figure 5.5: Phase inversion masks.

Figures 5.6 and 5.7 show results from experiments where image features were artificially blurred by horizontal motion blur kernels of varying length and had a relative offset of 55

pixels applied. In each experiment, the correlation S_{fg} is found between the unblurred original image and a copy with artificial motion blur. Figure 5.6 shows one line of samples from S_{fg} without a rectification mask. The double peak structure is clearly visible for blurs longer than 3 pixels, and peaks move further apart with increasing blur length. Figure 5.7 shows the phase correlation surfaces with the appropriate phase rectification applied. The peaks here are all clear above their relative noise floors, and have the correct coordinate.

The method of finding a mask based on the Fourier transform of the motion blur filter and using it in modified phase correlation, as in Equation 5.6, will be referred to as "velocity corrected phase correlation" (VCPC).

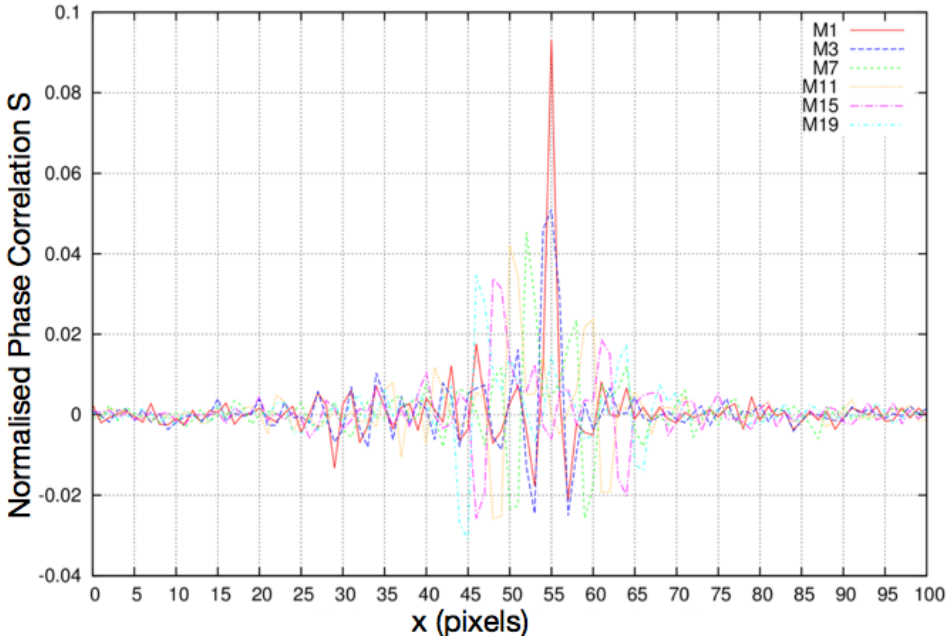


Figure 5.6: Image lines extracted from S_{fg} between an unblurred image and copies shifted by 55 pixels with motion blur of length indicated by M. For motion blurs greater than 3 pixels, there are two peaks. Figure: Tom Cox.

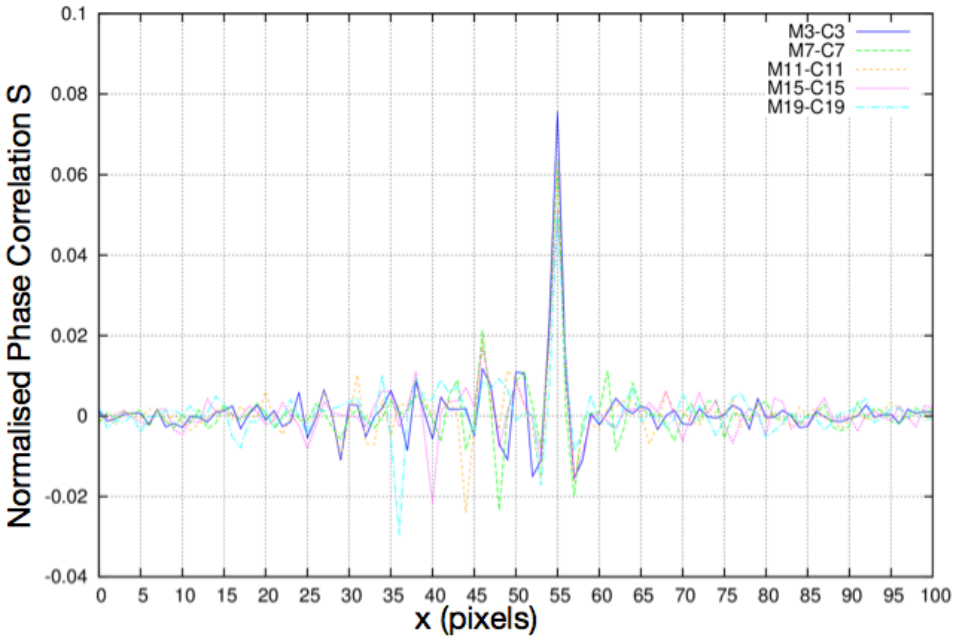


Figure 5.7: A row from phase correlation solution surfaces with appropriate rectification masks applied. The double peak structures visible in Figure 5.6 have disappeared. All the peaks are overlaid on the correct location. Figure: Tom Cox.

5.3.3 Estimating the Rectification Mask

In the examples described above we have calculated the rectification mask from the Fourier transform of the motion blur kernel. In real use, the motion blur may not be known *a priori*. Therefore we must derive some way of calculating a rectification mask for unknown blur.

If the direction is known (or can be estimated), then by assuming some motion blur length, and applying the appropriate motion mask, a strong indication of the correct motion mask can be found: A phase correlation S_{fg} has been found between an image and a copy of the same image with artificial motion blur of 7 pixels and a shift of 55 pixels. Figure 5.8 shows a row from this correlation where several different rectification masks m have been applied. The C-value in the legend indicates the length of blur used to generate rectification mask. In the case where the rectification mask is correct there is a clear peak at 55, as expected. In all the other cases, there are pairs of high peaks, each followed immediately by a low trough. The separation between these peaks is (approximately) equal to the magnitude difference between the actual motion blur and the motion blur used to generate to the rectification mask. None of these peaks are as high as in the case where the rectification mask matches the true motion blur.

The results in Table 5.1 suggest that a fast algorithm to estimate and match images where

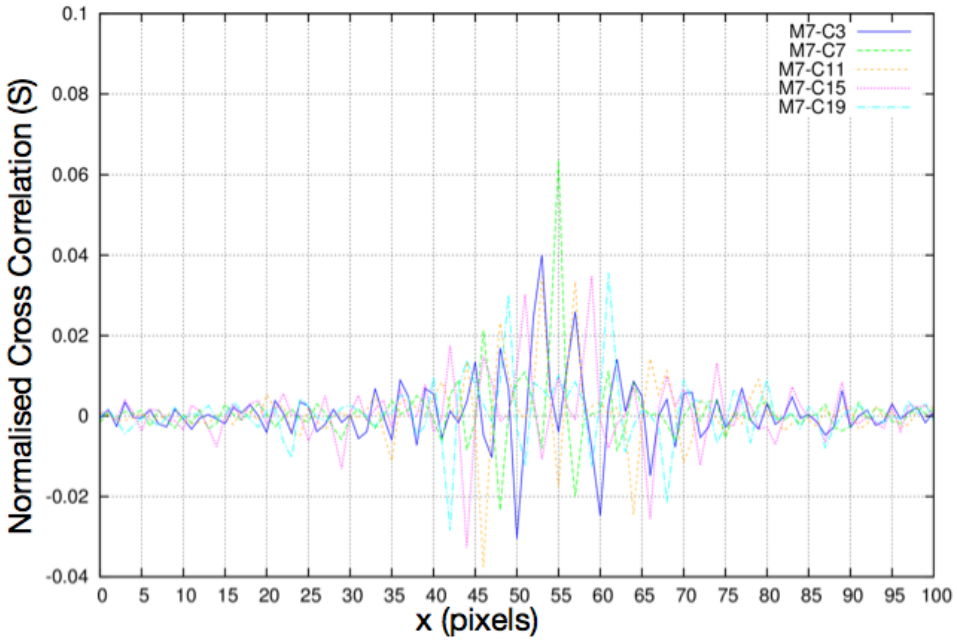


Figure 5.8: The effect of applying the wrong rectification mask to a motion blurred image. Figure: Tom Cox.

Table 5.1: The peak separation distance compared to the difference between the actual motion blur and the assumed motion blur.

Motion Blur M	Rectification mask R	$\lvert M - R \rvert$	Measured distance between peaks S
7	3	4	4
7	7	0	0
7	11	4	4
7	15	8	8
7	19	12	12

one suffers from motion blur may be possible. After one iteration an estimate of one (or two, if $R - S$ is positive) possible motion blurs is available. This estimate can be used to predict one or two new motion blur masks, one of which would be expected to give a single peak at the correct offset.

5.3.4 Sampling Limits

The rectification mask is constructed, either from the blur parameters, or the Fourier transform of the blur filter, as described in Sections 5.3.2 and 5.3.3. The mask is an array of the same size as the image it is applied to. Figure 5.5 shows the typical structure of the phase inversion masks. There is always a double-width band at the centre which corresponds to the central

lobe of the Sinc function. Beyond that central band there are a number of bands of equal width. The number of bands is directly proportional to the blur length, which implies for longer blurs, there are more bands.

In the continuous case the mask m is found using Equation 5.5. But a real implementation of this method requires a discrete mask. So the mask must be sampled somehow. The experiments described here use a sample and hold approach, where the value of the mask at the centre of the sampling pixel is chosen. This approach introduces some limitations.

Samples which are close to the boundary between bands represent a problem for this method. The left plot in Figure 5.9 shows the mask for a 16×16 pixel phase correlation with a motion speed of 5 pixels. Most of the samples in the mask are fully in either a "1" band or a "-1" band, but a few are in both. The right plot shows how this problem is compounded for a motion blur of length 13. Relatively few samples remain which are fully in one band.

It is worth noting that the motion blur filter magnitude in the vicinity of these band boundaries (which are just the zero-crossings of the sinc function) will be very small, so even a sample which is mostly in one band will suffer some significant attenuation. For the longer motion blur in Figure 5.9 many samples include both the edge of a band, and the centre of the same band.

The noise suppression strategy discussed in Section 5.3.6 will reduce the impact of the small-signal components to the overall analysis, but it is clear from Figure 5.9 that as the motion blur length approaches the patch size, very few well-defined mask samples remain.

Because of this, a drop off in performance is expected for motion blur above a particular length. This limitation was observed near the end of the work, and as such has not been thoroughly developed. The future work section contains ideas for further development and improvement of this method.

5.3.5 Squared Cross Power Spectrum Phase Correlation (SCPS)

Ojansivu and Heikkilä described a method for dealing with motion blur in phase correlation [19]. Their method is to square the cross power spectrum (strictly, raise to any even power, m), then halve (divide by m) the offset measured by finding the peak of the correlation S_{fg}.

$$S_{fg} = \mathcal{F}^{-1}\left(\frac{m\bar{F}G}{|\bar{F}G|}\right)^2 \tag{5.7}$$

Squaring the cross power spectrum means the phase of each component is doubled. Denoting the phase of a component each from images f and g as θ and ϕ, the corresponding component of the normalised cross power spectrum is $\phi - \theta$. Squaring the normalised cross power spectrum gives phase $2(\phi - \theta)$. Now, if either of these components has been phase

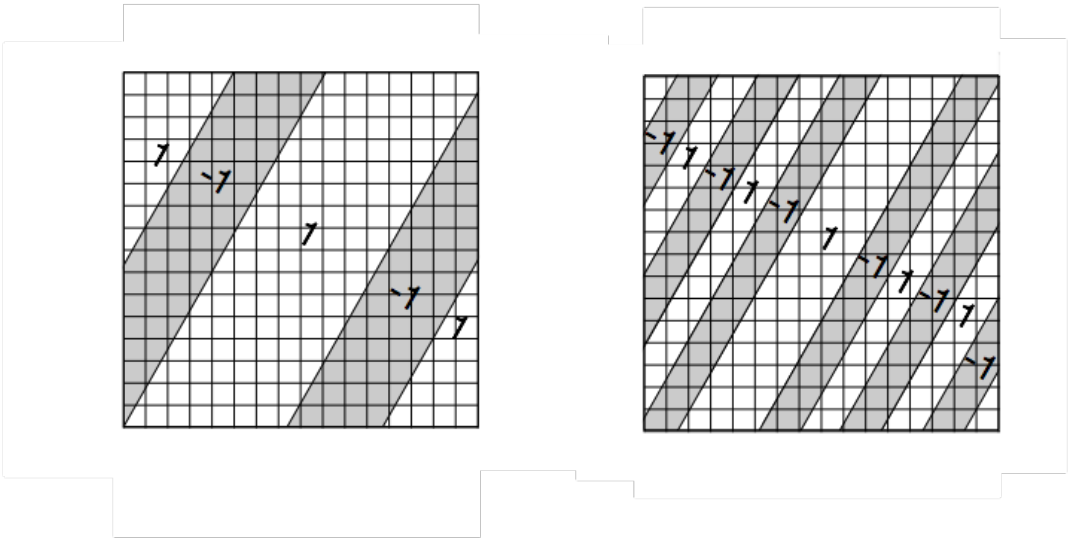


Figure 5.9: The impact of quantisation on masks for different motion blur lengths.

inverted as a result of motion blur, then the normalised cross power spectrum is $\phi - \theta - \pi$, and squared is $2(\phi - \theta - \pi) = 2(\phi - \theta) - 2\pi$. Since a phase angle is functionally identical when 2π is added or removed, squaring the cross power spectrum normalises the effect of phase inversion. The multiple of 2 can be dealt with by noting that phase angle varies linearly with image position *for a given frequency*. So scaling all phases by n results in the pixel offset also being scaled by n.

The key drawback of this method, when compared to VCPC, is that the maximum possible offset which can be found is $p/2n$, for an image region of size $p \times p$. VCPC is capable of finding the same range as ordinary phase correlation, up to $p/2$.

This method has a similar approach to VCPC, but is limited in that it can only determine the patch offset to within hald the patch size. Results are included below for this method alongside the proposed velocity correction method, and referred to as SCPS.

5.3.6 Dealing with Noise

Each frequency component will not necessarily contain useful information. The Fourier transform of many natural images is sparsely populated. Often many high frequency components in an image contain only noise. Usually the main source of noise in digital images is thermal noise arising directly from the sensor. This noise tends to be full-spectrum. So spectral components with little or no energy in the image will be dominated by noise. The fewer components which contain image information, the more the phase correlation result will be corrupted.

Random noise in phase will result in a poorer signal to noise ratio in S_{fg}. The normalisation step can be modified to suppress data which is likely to be noise.

Equation 5.6 becomes:

$$S_{fg} = \mathcal{F}^{-1} \frac{m\bar{F}G}{|\bar{F}G| + \alpha} \tag{5.8}$$

where α is some small value. This approach has been used in previous work on phase correlation within BBC R&D, although may not have been published. BBC Research Technical Report 1990-11 [88] mentions setting small components to zero, although thresholds are not mentioned. Using Equation 5.8, components of the cross power spectrum whose magnitude is comparable to α will have half the magnitude before the inverse Fourier transform step. As always when dealing with noise, there is no distinguishing a small wanted signal from a small noise signal. The fewer frequency components contain signal at a level much greater than α, the less effective this method, and the more poorly localized the peak in S_{fg}. A similar adaption is made for SCPS.

5.4 Timing

VCPC and SCPS are quite similar from a computational standpoint. The key difference is the computation of the mask. However, this is not an onerous additional requirement. As will be shown, it is equivalent to computing an additional feature per frame. If there are many features in the frame, then computing an additional one is not significant. If there are few features per frame, then the method is fast enough for this to not impact the real time requirement.

5.4.1 Complexity analysis

Both methods exist within a framework where pairs of features are presented, and the resulting registration is returned. Therefore this analysis will look only at the operation of VCPC and SCPS within this framework. Table 5.2 contains an analysis of both algorithms using "big-O" notation. Here, n is the number of samples in an image patch. Clearly the theoretical complexity of both algorithms are the same, in this framework.

If the mask is not known, it can be found by doing VCPC on a pair of features and finding the distance between the two largest peaks, as described in Section 5.3.3. This can be computed in the same time as VCPC given in the table. (Finding the two highest values in an array takes the same time as finding the highest value.)

Table 5.2: Complexity of phase correlation based algorithms

Task	VCPC $\mathcal{O}$	SCPS $\mathcal{O}$
FFT of patches	$n \log n$	$n \log n$
multiply spectra	n	n
multiply by mask	n	—
square CPS	—	n
normalise	n	n
inverse FFT	$n \log n$	$n \log n$
find peak	n	n
Total	$n + n \log n$	$n + n \log n$

5.4.2 Suitability for real time use

Time was not available duing the course of this work to implement an optimised real time version of VCPC or SCPS. However, some simple experiments on an ordinary laptop computer suggest that an algorithm with complexity $\mathcal{O}(n + n \log n)$ is suitable for real time feature matching. The parts which run in $\mathcal{O}(n \log n)$ are Fast Fourier Transforms. Using the FFTW library [89] via the Julia programming language, an FFT on a 32×32 array of 64 bit floating point numbers is computed in about 16 microseconds on a 2.9GHz laptop running on one core. The benchmarks published on the FFTW website report 8 microseconds for the same FFT on an older processor at 3.0GHz. The discrepancy is likely due to the optimised conditions for the benchmark, and the overhead in the Julia wrapper to allow straightforward calling into the FFTW library.

Assuming that the $\mathcal{O}(n \log n)$ parts of the algorithm take about 30 microseconds (for one FFT and one inverse FFT), and the others take less than that, there is time in a 20 millisecond frame to compute several hundred VCPC or SCPS matches. In Section 1.1.1 a strategy is discussed for limiting the number of image features by ensuring no image region is too densly populatted. If VCPC or SCPS is implemented as part of a system which also limits the number of features per frame to less than a few hundred, then either method would be suitable for real time tracking.

The overhead arising from computing the mask is equivalent to computing at most two additional matches. The number of matches can vary quite significantly in real use, and is likely to be capped in a system where real time performance is a strict requirement. In all the systems discussed so far, the cost of computing two matches is at most two percent of the total run time. Therefore, the overhead will be practically insignificant in all real use cases.

5.5 Results and Discussion

This section presents results for template matching precision, using the experimental framework described in Chapter 4.

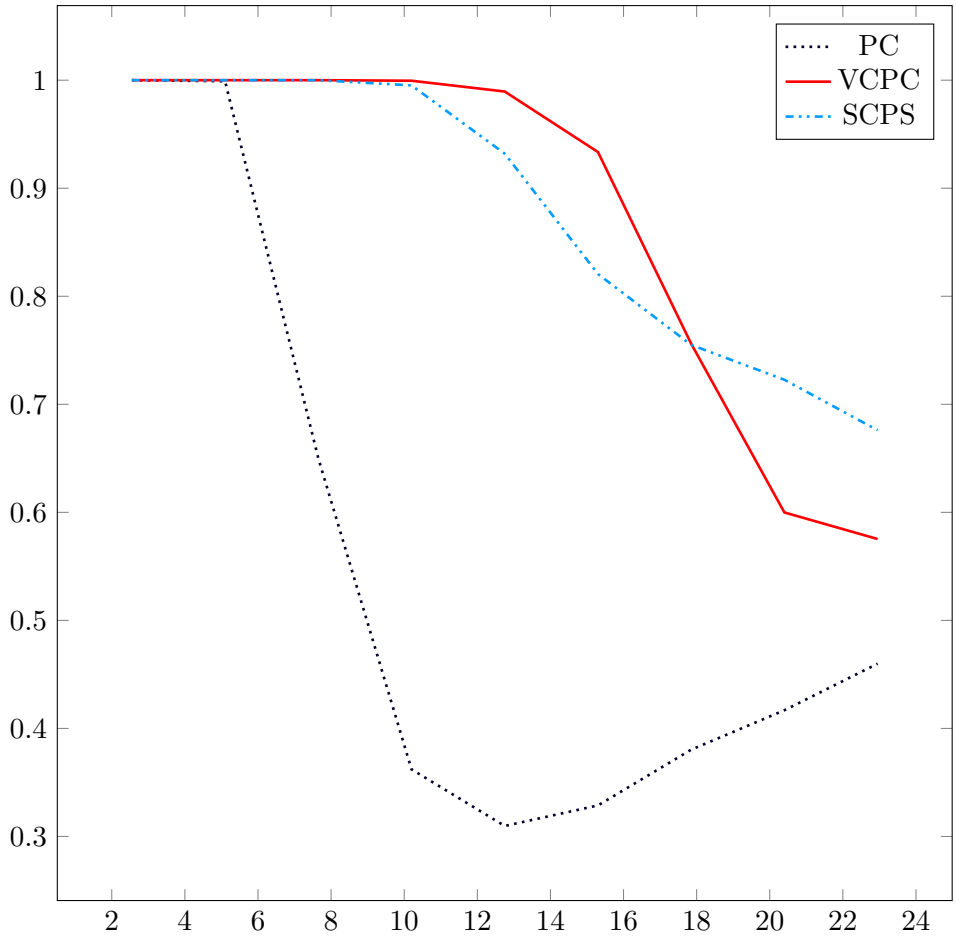


Figure 5.10: Mean phase correlation result across all targets

Figure 5.10 shows the mean results of reference tracking features on all sequences. Figure 5.11 shows the results from two individual targets. In all cases, patches of 32×32 pixels are matched between images. Identical feature locations are used. The figures are the mean precision for the whole sequence, or sequence indicated.

The precision achieved by VCPC clearly beats SCPS for longer motion blurs in the average case (Figure 5.10.) Figure 5.11 shows how the improvement over SCPS varies between marginal and significant, depending on the image content.

Figure 5.10 also shows the drop off in performance resulting from longer motion blurs, and sampling effects in the masks, as discussed in Section 5.3.4.

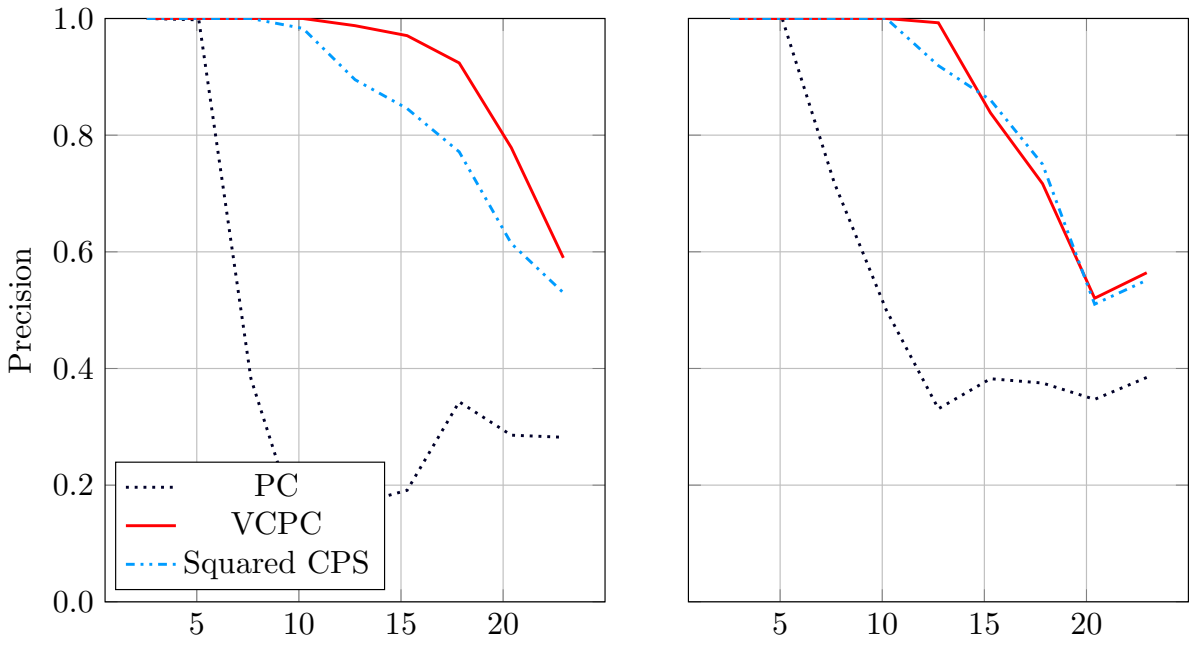


Figure 5.11: Results for the targets *bricks* and *paris*, showing the content-dependence of the VCPC method.

5.6 Conclusions

The results of these experiments show that, for long motion blurs of up to 18 pixels, VCPC outperforms SCPS for matching blurred to non-blurred features, and gives equally good performance for shorter blurs. The timing analysis in Section 5.4 shows that the execution time of the methods is comparable. As a side effect of using this VCPC, the motion blur parameters are available at trivial extra computational cost, which OHPC does not produce. Finally, VCPC is capable of detecting offsets of up to half a patch width, rather than half the patch size, which is the limit with SCPS.

Chapter 6

Velocity Corrected SIFT

6.1 Introduction

This chapter proposes a number of novel modifications to SIFT feature matching. These are all based on observations about the impact of motion blur on SIFT features. These approaches are assessed and compared using the experimental framework described in Chapter 4. One of the methods, called Speculative vector space flattening, is found to provide a significant improvement over matching ordinary SIFT features, where one image has motion blur and the other has not. All methods introduced introduce a small additional computational overhead, although not enough to render this method unusable in real time.

This chapter is structured as follows; Section 6.2 summarises the operation of SIFT feature description and matching, and how motion blur affects both. Section 6.3 proposes several methods to make SIFT matching more tolerant to motion blur, and Section 6.4 explains the experiments conducted on both simulated and real video sequences, with the results being presented in Section 6.5. Section 6.6 contains conclusions and some discussion of future work.

6.2 Background

SIFT descriptors were discussed in Section 2.7.1. A SIFT descriptor is a representation of a region of the image around a feature point. The size of this region is determined by the characteristic scale determined by the detector. (Or can be fixed by the operator.) The region is divided into a grid of squares, which can be oriented with a characteristic direction from the feature detector, or can have fixed orientation with respect to the image ("upright" SIFT features). Within each square, a weighted histogram of gradient directions is computed. The weight is a Gaussian with a peak at the feature point and σ of half the size of the region selected around the feature point.

The energy from each gradient sample is spread around the histogram slightly, to avoid abrupt changes in histogram values if the feature point is not localized precisely. Figure 6.1 illustrates the effect. In the left panel, the SIFT descriptor without the energy spreading effect is shown. The dot and yellow line indicate the location of a gradient sample. The red line is the quantised distribution of that gradient sample into the nearest bin, at the nearest angle. In the right panel, the contribution from the same gradient sample is distributed to bins adjacent in angle, and in both image directions. Each sample from the gradient image is weighted by $1 - R$, where R is the distance from the histogram bin centre — be that in angle or (scaled) pixels.

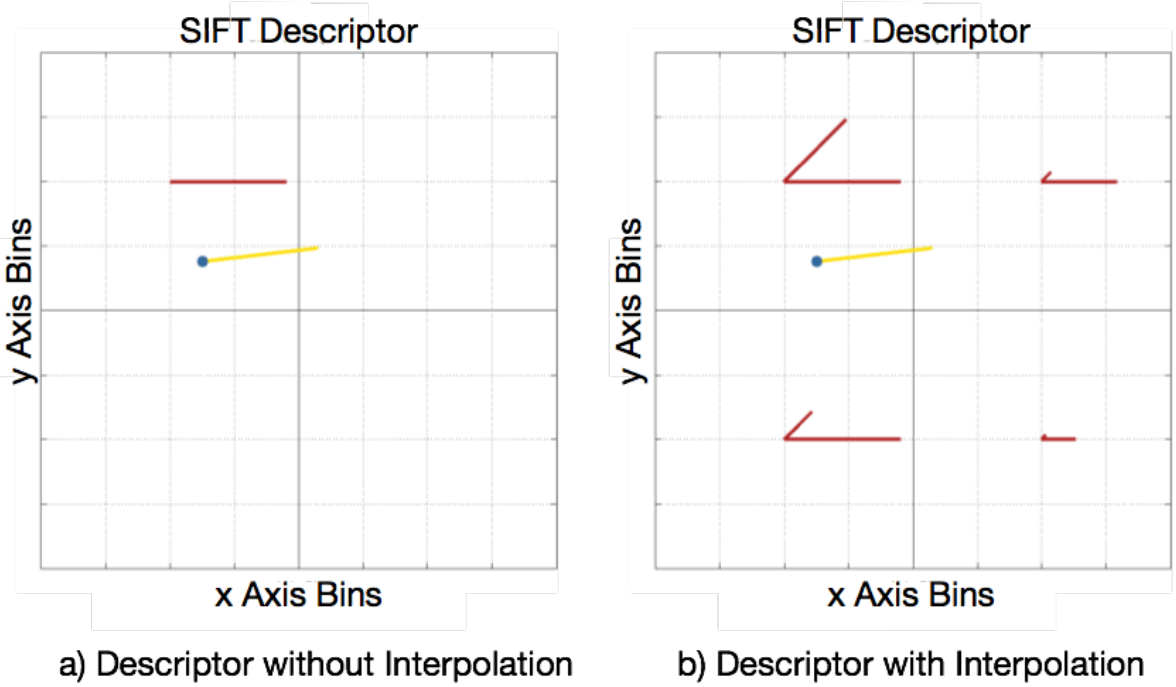


Figure 6.1: Distribution of gradient energy within a SIFT feature. Figure: Neill Campbell.

The histograms are concatenated together to form the descriptor "vector". (Whether or not the SIFT descriptor is a vector with a meaningful vector space is a moot point. Treating it as such has some benefits, as shall be seen.) The vector is normalised so it has unit length, then to prevent large values from dominating, any single coefficient which is greater than 0.2 is reduced to 0.2, and the vector is unit-normalised once more.

In [20], Lowe proposes a 4×4 grid of squares, and within each an 8-bin histogram. This gives a 128-coefficient vector of gradient values. These parameters are used here.

In broadcast video and feature films rotation invariance is not important because the camera generally does not roll. Therefore "upright" SIFT features are used in the methods described

below. Indeed, these methods rely upon there being a well-defined range of image directions for each histogram bin.

6.2.1 SIFT Feature Matching

Conventionally the matching score between two SIFT features d is computed by treating the feature vectors as describing points in a Euclidean vector space, and finding the distance between them. This is expressed as the Euclidean norm:

$$d^2 = \sum_{i=0}^{127} (A_i - B_i)^2$$

Where A and B are SIFT descriptors, and A_i is the ith coefficient in A.

The impact of motion blur is a disproportionate increase in distance between features A and B in feature space, as the Euclidean norm emphasises large coefficient differences over small ones.

SIFT Feature matching is a real time operation, as discussed in the literature review. For the purposes of comparison, the steps to compute the difference between a normal pair of SIFT descriptors is given here. This will be revisited for each modification, to see the additional work required. Table 6.1 shows the steps needed to compute the difference between a pair of SIFT descriptors. This gives a total of 383 simple operations, readily parallelisable, and one complex operation.

Table 6.1: Steps to compute a SIFT difference and their timing.

Logical step	Work required
Find differences $A_i - B_i$	128 subtractions
Square each difference	128 multiplications
Sum the squared differences	127 additions
Square root the result	1 square root

6.2.2 Motion Blur and SIFT Descriptors

In Chapter 3 it was shown that a moving edge is strongly affected by motion blur. Recall that an edge, with constant velocity across an image sensor parallel to the gradient direction, will be recorded as an intensity ramp with a small constant gradient. The gradient of the edge is a large value, tightly localized in the image. The gradients of the ramp and the edge will result in very different contributions to SIFT descriptors. A simple thought experiment will show that an edge moving perpendicular to the gradient direction is not affected, and its gradient is consequently unchanged.

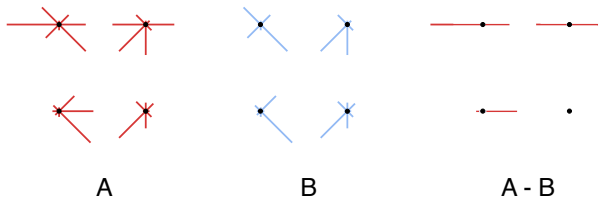


Figure 6.2: Hypothetical impact of horizontal motion blur on a SIFT feature. The black dot indicates the centre of the histogram.

How should this direction-dependant effect on gradient be interpreted in SIFT descriptors? Begin by assuming that edge directions perpendicular to to the direction of the motion blur will be the only ones affected. Figure 6.2 shows a toy version of this idea. A and B are SIFT responses to a particular image region, where the image for B has been subject to some horizontal motion blur. The bins of B in the horizontal directions are small because they represent response to vertical edges, which have been reduced in intensity and dispersed by the motion blur. In the difference histogram A-B the horizontal bins contribute large differences, whereas differences elsewhere are small (zero, in the illustration). In this figure, small histogram coefficients, ie short bars, represent a better match than longer bars. These large differences cause two features differing only in motion blur to have a large inter-feature distance.

Note that the bottom-right histogram in Figure 6.2 has no large differences — if there is no energy in the histogram in that region, it cannot be attenuated, and there will be no (or little) mis-match.

This model is likely to be too simple. In general, the impact of motion blur is likely to be strongest for edges perpendicular to the direction of blur, and weakest for edges parallel to the blur, with the effect varying smoothly for directions in-between. SIFT features accumulate edge information into histogram bins, so the bins which include the edges perpendicular to the blur will be most strongly affected, and those bins which include edges parallel to the blur will be least affected. The methods described in Section 6.3 attempt to deal with this distortion.

6.2.3 RootSIFT

In [90], Arandjelović and Zisserman point out that in texture classification and image categorization, it has been found that the Hellinger measure or χ^2 measure provide better results when comparing histograms than the Euclidean distance. Observing that SIFT descriptors are histograms, they propose RootSIFT matching: The values in the SIFT descriptor are modified such that finding the Euclidean distance between two RootSIFT descriptors is equivalent to

comparing the original SIFT descriptors using the Hellinger kernel.

RootSIFT features are created from SIFT descriptors by first normalising the coefficients such that they sum to one, then taking the square root of each coefficient. Once the conversion is done, RootSIFT features can be used in place of SIFT features throughout the feature matching pipeline. Arandjelović and Zisserman report a 12% increase in performance for an object retrieval task. This general-purpose method for improving the performance of SIFT features will be compared to the methods proposed in this chapter. (They are both relatively low-overhead, which makes the comparison useful.) Because RootSIFT is a drop-in replacement for SIFT features, results are also presented for the proposed modifications built on top of RootSIFT features.

6.3 Proposed methods

Four modifications to SIFT descriptor matching are proposed in this Section. All attempt to exploit the properties of motion blur and SIFT descriptors, in different ways. The methods are:

- Directional Weighting.
- Naive Vector Space Flattening
- Directional Vector Space Flattening
- Speculative Vector Space Flattening

The Directional methods require that the motion blur direction is known. Naive and Speculative Vector Space Flattening do not. In some circumstances an estimate of the motion blur vector will be available (for example, during the tracking of a pan tilt zoom camera.) Other times it will not. In those cases, some approach to estimating the blur vector will be required. The literature contains some possible approaches, including [91, 92, 93, 94, 95, 77].

6.3.1 Directional Weighting

A simple change, given the impact of motion blur described above, is to weight the contributions of the coefficient differences according to which are most likely to have been affected by a given direction of blur. Coefficients corresponding to gradient directions perpendicular to the motion blur direction are downweighted. Weightings are computed using a sinusoid. This was found to be a good approximation of average amount by which coefficients are altered by motion

blur. The weights w_i are calculated using:

$$w_i = 0.5 + |\sin(\theta_i - \theta_b)|$$

Where θ_i is the central angle of the ith SIFT histogram bin, and θ_b is the direction of the motion blur.

Using this method the matching score d between two SIFT features is

$$d^2 = \sum_{i=0}^{127} w_i (A_i - B_i)^2. \tag{6.1}$$

Table 6.2: Steps to compute a directionally weighted SIFT difference and their timing.

Logical step	Work required
Apply weights	128 multiplications
Find differences $A_i - B_i$	128 subtractions
Square each difference	128 multiplications
Sum the squared differences	127 additions
Square root the result	1 square root

Table 6.2 gives the steps to compute a directionally weighted SIFT difference. The computation of the weight vector will be an additional overhead. For the appications described in Sections 1.1.1 and 1.1.2, the same motion blur vector is used everywhere, so this will be a small overhead compared to matching around one hundred image features. This method represents an increase of about one-third in computation time over ordinary SIFT.

6.3.2 Naive Vector Space Flattening

In Section 6.2.2 it was postulated that the impact of motion blur on SIFT descriptor matching was to introduce a small number of much larger coefficient differences. These Vector Space Flattening methods are different attempts to discard these larger differences, without impacting the rest of the matching process. The vector space in which inter-feature distances are measured can be *flattened* by removing some of the dimensions. A new Euclidean distance between the descriptors is found in this new vector space. Motion blur direction information is not required for this method.

Define D to be the vector of coefficient differences:

$$D_i = A_i - B_i \tag{6.2}$$

Naive Vector Space Flattening finds and then discards the largest n entries in D. The Euclidean norm of the remaining values in this shortened $\hat{D}$ is the matching score.

Table 6.3: Steps to compute a naive vector space flattening difference and their timing.

Logical step	Work required
Find differences $A_i - B_i$	128 subtractions
Sort differences	$\mathcal{O}(128 \log 128)$
Square each difference	120 multiplications
Sum the squared differences	119 additions
Square root the result	1 square root

Table 6.3 gives the timing for naive vector space flattening. The main differnce is the sorting step, which will dominate the run-time. Since $\log_2 128$ is 7, the sorting step will take at least 7 times as long as ordinary SIFT matching. However, since ordinary SIFT features are computable in thousands per frame, in real time [20], reducing this count by a factor of 7 is likely to still result in real time performance.

6.3.3 Speculative Vector Space Flattening

In a tracking or visual search system, the final part of the SIFT descriptor matching process is to compare the matching score for a number of features in a small neighbourhood. If the best match is not less than 0.8 times the second best match, then it is assumed to be a false match. A possible side effect of Naively flattening the vector space of feature pairs which are not a match is to reduce the size of the second best match, thereby reducing the number of true matches. Speculative Vector Space Flattening is introduced to deal with this.

This method speculates that a large improvement in matching score might be gained by Vector Space Flattening. If the improvement is not sufficiently large, then matching reverts to ordinary SIFT: Speculative Vector Space Flattening proceeds as follows: The largest n entries in D are discarded. Now, if the Euclidean norm of $\hat{D}$ is less than t times the Euclidean norm of D, then $\hat{D}$ is the matching score. Otherwise, D is the matching score.

Table 6.4: Steps to compute a speculative vector space flattening difference and their timing.

Logical step	Work required
Find differences $A_i - B_i$	128 subtractions
Sort differences	$\mathcal{O}(128 \log 128)$
Square each difference	120 multiplications
Sum the squared differences	119 additions
Square root the results	2 square roots
Decide which value to use	1 comparison

Table 6.4 gives the timing for speculative vector space flattening. As with naive vector space flattening, the main extra work is in the sorting step. As with that method, this is likely to still be suitable for real time implementation.

6.3.4 Directional Vector Space Flattening

If the direction of the motion blur is known, then the SIFT coefficients likely to be worst affected can be determined and removed. This can be thought of as a binary weight. If the histogram directions are indexed by α_i, and the bin containing the blur direction is $alpha_b$, then

$$w_i = \begin{cases} 1 & \text{if } \alpha_i \neq \alpha_b \\ 0 & \text{if } \alpha_i = \alpha_b \end{cases}$$

The matching score can then be found using Equation 6.1.

Directional Vector Space Flattening also includes the "Speculative" final comparison step to improve performance when a feature point has multiple similar matches.

Table 6.5: Steps to compute a directionally flattened SIFT difference and their timing.

Logical step	Work required
Apply weights	128 multiplications
Find differences $A_i - B_i$	128 subtractions
Square each difference	128 multiplications
Sum the squared differences	127 additions
Square root the result	1 square root

Table 6.5 gives the timing for directional vector space flattening. Because this can be computed as a weight vector, the overhead in timing is similar to directional weighting. It is suitable for real time implementation.

6.3.5 Implementation

The implementation of all of these methods is straightforward. These experiments used the VLFeat [96] SIFT implementation to compute SIFT descriptors. Directional Weighting adds additional complexity at $\mathcal{O}(n)$, where n is the number of SIFT features per frame, if it can be assumed that the estimate of the velocity for a particular feature does not change. All Vector Space Flattening methods add additional complexity at $\mathcal{O}(m^2)$, where m is the average number of matches tested. (The length of the sort is always 128 at most, so the sort is always in constant time. There are m^2 feature comparisons per frame.)

The proposed modifications to SIFT matching were implemented as part of a test harness in the Julia programming language [97].

6.4 Experiments

An experiment was conducted to determine appropriate parameters for the Speculative and Naive vector space flattening methods.

6.4.1 Parameter setting

An experiment was carried out to determine a good estimate for n for speculative and Naive vector space flattening, and t for speculative vector space flattening. Clearly the optimal n and t will always be image dependent, but this cannot be known *a priori*, so it must be estimated.

A test data set was created using a set of images with diverse content. Each image had synthetic motion blur and noise added to create pairs of blurred and non-blurred images. Harris corners [13] were detected on a 2-times downsampled version of the non-blurred image in each pair. Upright SIFT descriptors are computed at the feature points in both original and distorted images. Matching scores are computed between each feature descriptor from an original image and each feature descriptor within 50 pixels of that feature in the blurred image, using Speculative Vector Space Flattening.

If the best matching score is less than 0.8 times the second best, then the feature point is assumed to have been correctly matched. (The 0.8 factor was originally proposed by Lowe [20].) Otherwise the point is assumed to be a false match.

The 23 images from the Kodak Sampler Photo CD [1] are high quality images with diverse content. They were chosen as the basis for a set of image pairs. Figure 6.3 shows the results of the experiment to vary n, aggregated over all image pairs. If c_{SIFT} is the number of features correctly matched with SIFT, and c_N is the number correctly matched with some n, the improvement metric is $(c_N - c_{SIFT})/c_{SIFT}$. From the Figure, $n = 8$ gives the greatest improvement.

Figure 6.4 shows the result of the experiment to vary t. From the figure, $t = 0.8$ gives the greatest improvement. These values for n and t will be used in the experiments in this chapter.

[1] Retreived from `http://www.math.purdue.edu/~lucier/PHOTO_CD/`

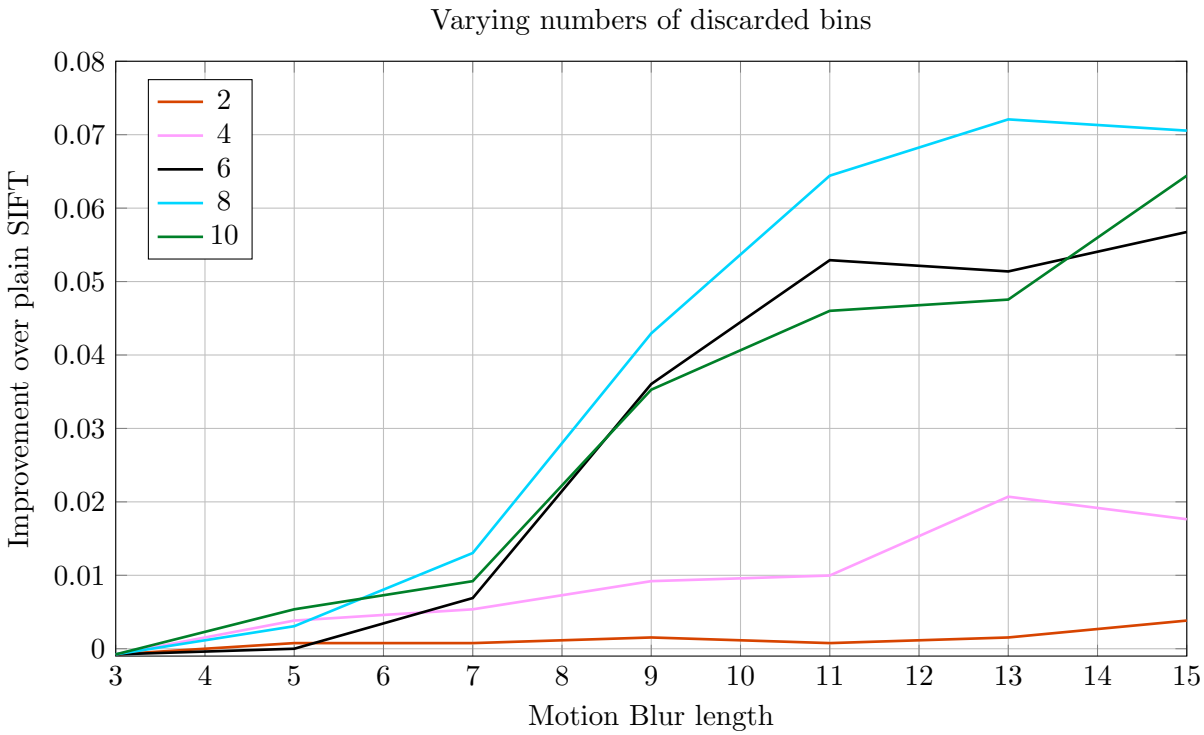


Figure 6.3: Proportional improvement for varying n bins discarded. Line colour indicates n.

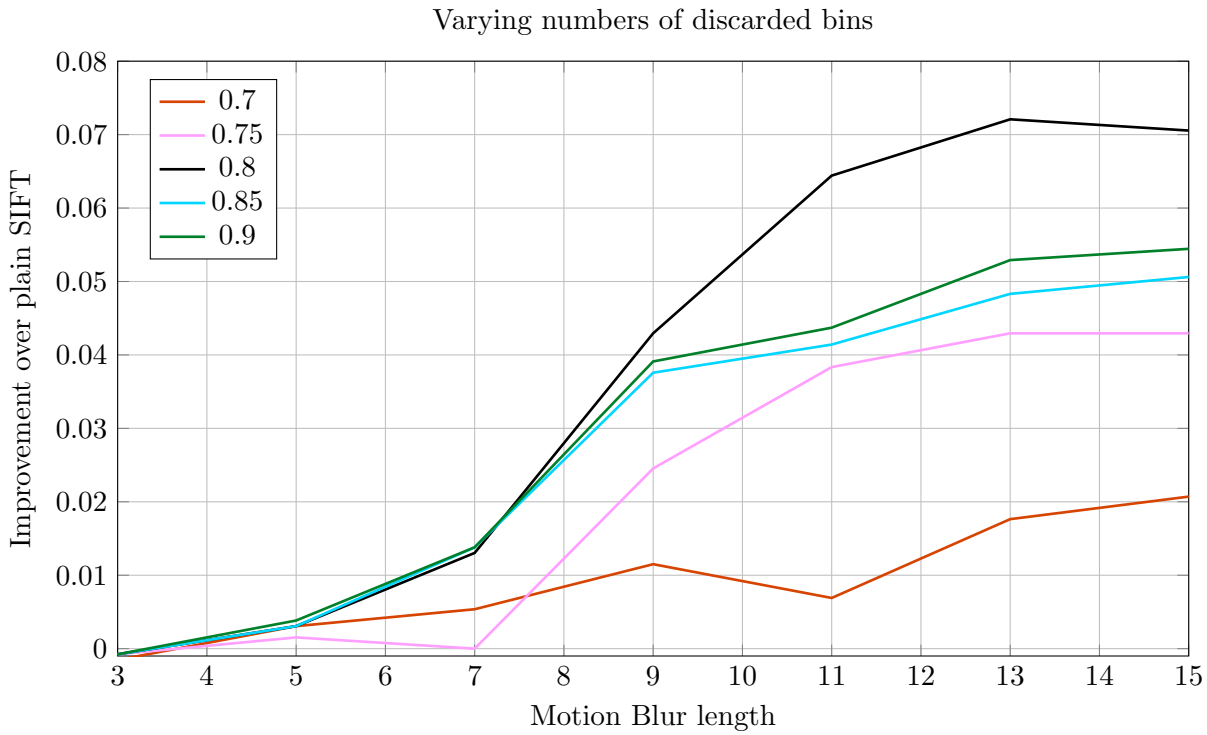


Figure 6.4: Proportional improvement for varying the improvement threshold t. Line colour indicates threshold.

6.5 Results

Results are given here showing precision for each of the methods and for each of the tracking targets used, following the procedure described in Chapter 4. The aggregate precision is also shown, as well as some results using RootSIFT features.

Results for Reference tracking are shown first. Figures 6.5 and 6.6 show the precision of matching for all methods, plus ordinary SIFT, for each of the tracking targets individually. (In the legends, Vector Space Flattening is abbreviated VSF.) The left plot in Figure 6.7 shows the aggregate precision when the data from all tracking targets are combined. The right plot in Figure 6.7 compares ordinary SIFT and Speculative Vector Space Flattening computed on normal SIFT and RootSIFT features.

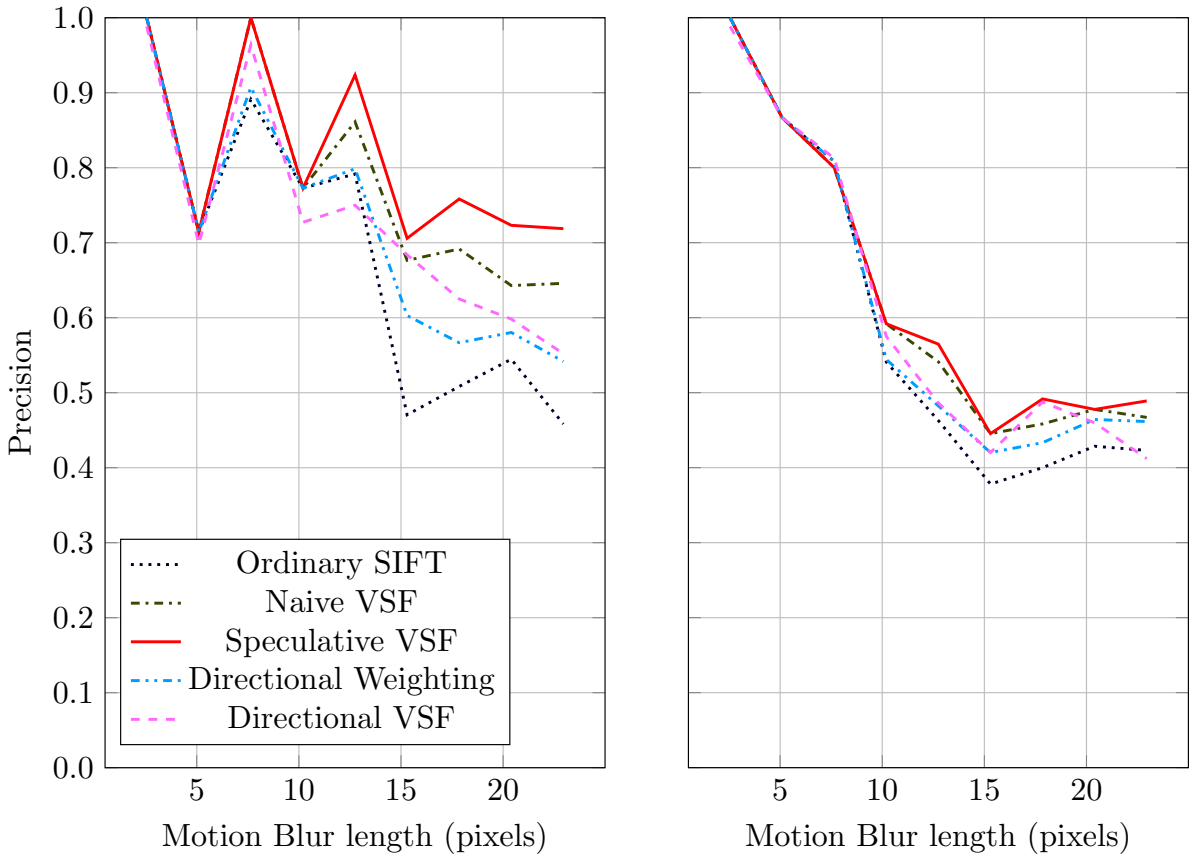


Figure 6.5: Feature matching precision. Results for the *mission* target is on the left, and *paris* on the right.

Directional weighting improves slightly over SIFT in each of the different target images, but is usually outperformed by the other methods.

In general, Directional Vector Space Flattening improves performance compared to SIFT for short motion blurs, but then does not do as well for longer motion blurs. The performance

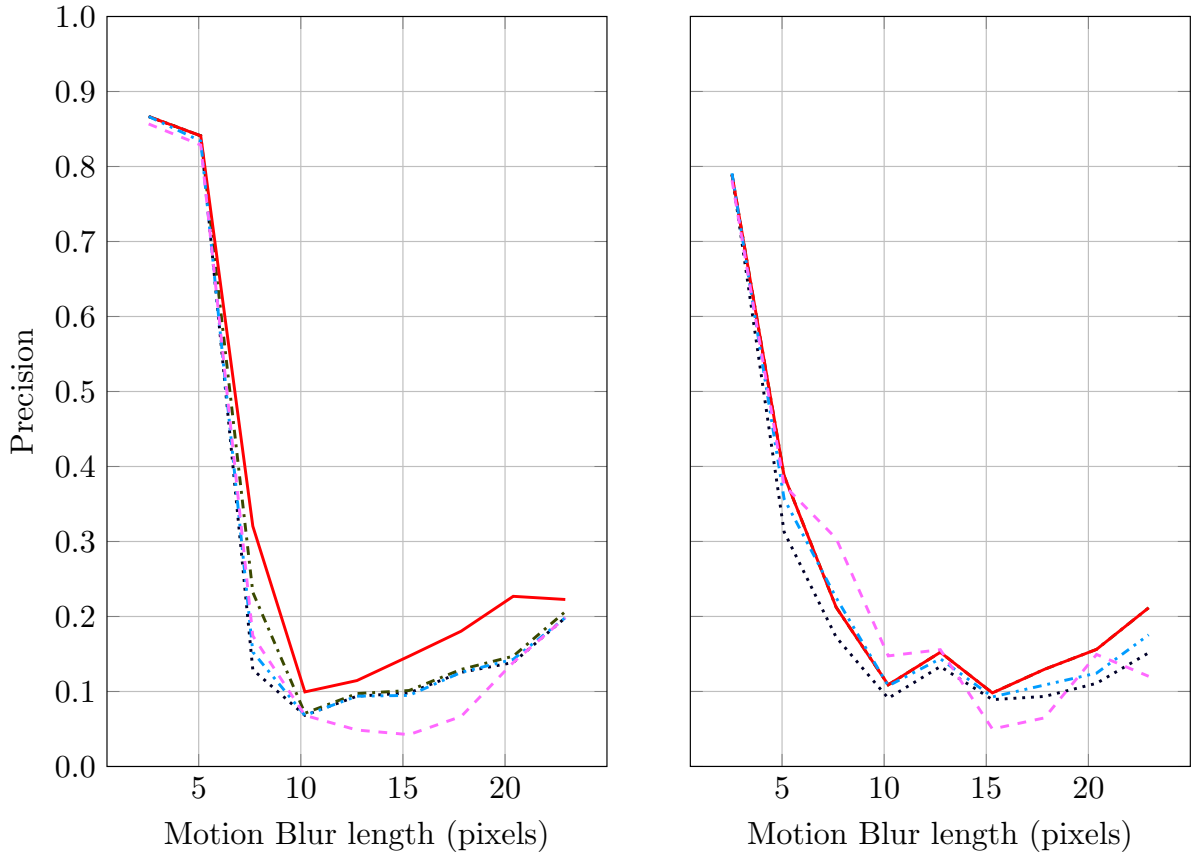


Figure 6.6: Feature matching precision. Results for the *bricks* target is on the left, and *building* on the right. Legend as in Figure 6.5

improvements for motion blurs of less than ten pixels are significant, particularly on the *building* image. Naive Vector Space Flattening usually provides a small improvement over SIFT.

Speculative Vector Space Flattening performs best overall. It has the best performance for *mission* and *bricks*, and is best for long motion blurs on *building*. On *paris*, there is a small performance regression for short blurs. This is not very important, as there will generally be more features available for shorter motion blurs. On the aggregate result plot in Figure 6.7, there is an improvement of about 10% in precision for longer motion blurs.

In Figure 6.7 RootSIFT shows a small improvement in matching for some blur lengths, but the overall impact is small compared with the improvement from Speculative Vector Space Flattening, particularly for longer motion blur.

Figure 6.8 shows the results for *Inter-frame* tracking. Clearly, plain SIFT works well in this case, and the improvements are small. Directional Weighting shows poorer performance than SIFT here because the method assumes that there is some difference in motion blur between the the features. This is the same result found by Gauglitz et al.

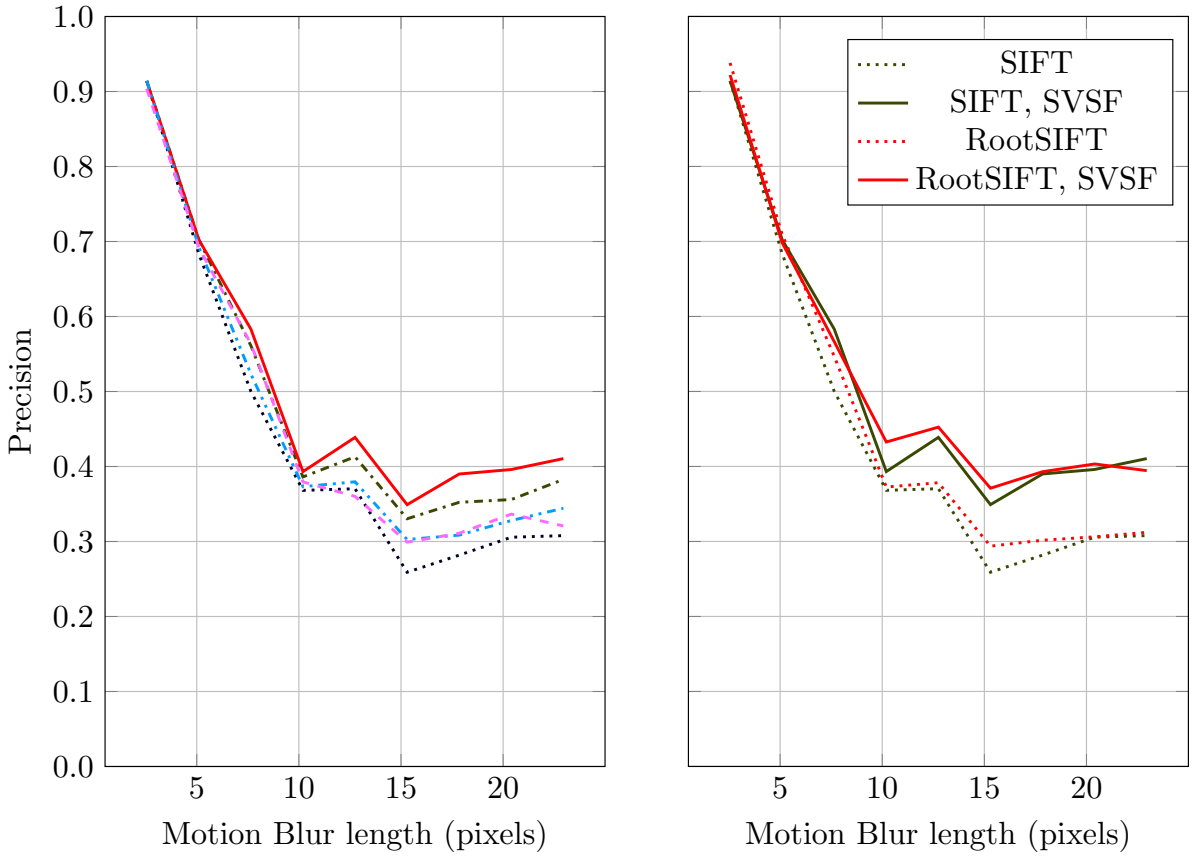


Figure 6.7: (left) Feature matching precision aggregate scores for all targets, (right) comparison of SIFT features and RootSIFT features.

6.6 Discussion

Of all the methods tested, Speculative Vector Space Flattening provides the most significant improvement over plain SIFT when matching unblurred features to those corrupted by motion blur. In an experiment matching an unblurred image with one containing 20 pixel long motion blur, on the *bricks* sequence, SVSF was able to correctly match 60% more SIFT features than ordinary SIFT matching. In the left panel of Figure 6.7 the average results across all textures show that for long motion blurs, SVSF is able to increase the number of correctly matched features by 30%.

SVSF has the useful feature that it is not required to know any of the characteristics of the motion blur, which can be difficult and time consuming to estimate blindly. It is worth examining why removing the n largest coefficient differences was more effective than removing those coefficient differences indicated by the simple model (In Section 6.2.2). Two factors taken together produce a plausible explanation:

One: The model suggests that only edges near-perpendicular to the motion blur direction

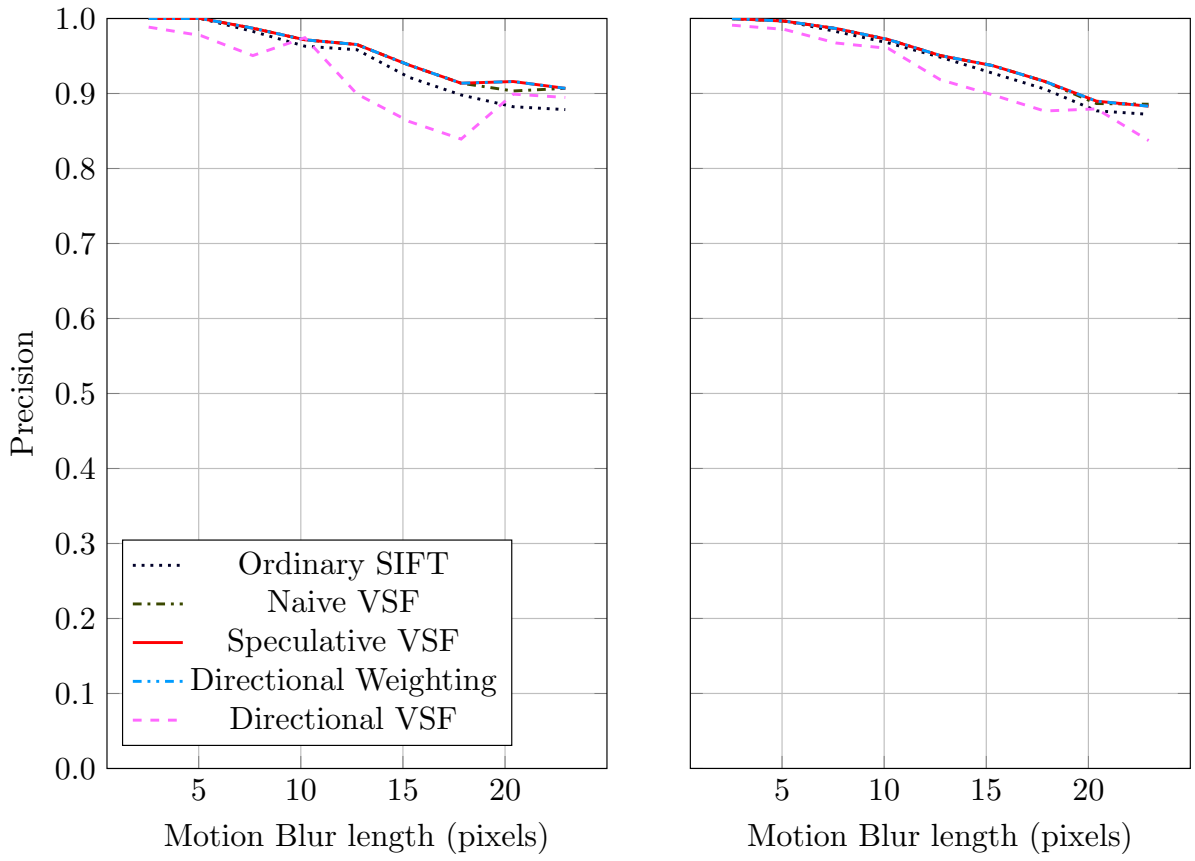


Figure 6.8: Inter-frame tracking. Left shows the results for *bricks*, and right is the aggregate.

will have their SIFT response altered. As noted earlier this is probably an over-simplification. It is likely that the image gradient in other directions will be affected as well.

Two: Stronger edge responses have a disproportionately larger effect when they disagree between two features than when they agree: Consider two edge responses. One is large, the other is small. Assuming they are not affected by any distortions, the difference between the small edge response in image 1 and the same small edge response in image 2 will be zero. The same is true of the two large responses. If some distortion causes these edge responses to be reduced in intensity by half in image 2, then the difference signal will now be in proportion with the size of the responses. (This is possibly related to the observations about Euclidean distance being poorly suited to comparing histograms, as noted in the motivation for Root SIFT. See Section 6.2.3 and [90].)

The intersection of these two effects will be a content-dependant effect: When the largest edge response differences are in the bins eliminated by Directional VSF, then it will be effective. If the largest edge response differences end up in other bins, then Speculative VSF will be more effective, and Directional VSF will have less effect, in proportion to the coefficient differences

removed.

Getting a low matching distance between the two correct descriptors is one result of this effect. A corresponding result will determine to what extent the other descriptors in the search area are likely to receive an even lower matching score.

Figures 6.5 and 6.6 bear this out. The difference in Precision score between Ordinary SIFT and Directional VSF shows more variance than the difference in Precision between Ordinary SIFT and Speculative VSF.

The implementation is very simple, and incurs a very small overhead beyond matching SIFT features conventionally. Although, if an application involves a very large number of features then the overhead will become large. Still, the increase in computational load is likely to be far smaller than any comparable methods such as [43, 44, 54, 98, 76, 2].

6.6.1 Future Work

These experiments report results on Harris features, which are found only on one scale of image. The detector which is described along with SIFT in [20], or some of the other described in Chapter 2 can detect characteristic scale. If both feature scale and motion blur length were known, then this could be taken into account when deciding whether to apply these methods or not.

The principle behind Speculative Vector Space Flattening might be applicable to other feature matching methods. Testing the principle with SURF features [30], Dual-Tree Complex Wavelet Transform-based features [99, 2] and others might yield useful results. It would also be worth testing in the presence of other image distortions, or a combination thereof.

This work has only investigated the impact on upright SIFT features. For applications where image rotation is a consideration, Speculative Vector Space Flattening may be worth investigating, with the caveat that orientation assignment is likely to suffer significantly in the presence of motion blur. Still, the results in this context would almost certainly provide further insight.

Speculative Vector Space Flattening relies upon high quality feature detection in the presence of motion blur. Figure 10 in [1] shows that most feature detectors do not reliably detect the same features, particularly when compared to a non-blurred reference frame. The better feature detectors get, the more the improvements will be gained from Speculative Vector Space Flattening.

Chapter 7

Conclusion

This Thesis posed two questions:

- Can the models describing motion blur be verified by experiment?
- Can the effect of motion blur be predicted and corrected for in the image feature matching process?

In particular, improvements were sought to match image regions containing motion blur with those which contain no blur in real time. The literature shows that matching image features containing similar motion blur already works well. The literature also contains examples [44, 45] of methods which perform well at matching blurred features to non-blurred features offline. The experiments described in this Thesis were concerned with real time performance.

The measurements in Chapter 3 showed that motion blur behaves in accordance with the models. By measuring the length of motion blur, it was possible to extract an estimate of exposure time which agreed with the camera settings up to one standard deviation with all but one of the cameras under test. These experiments also highlighted that motion blur is coupled with the effects of the optical parts of the camera. For high-precision image analysis, this coupling cannot be ignored.

Image noise was a source of bias on the estimated measurements of exposure duration. For some cameras this was not a significant problem, but the pictures from the For.A camera were sufficiently noisy to cause large over-estimates of the exposure duration. The effect of the point-spread function of the optical parts of the cameras resulted in lower precision results than would have been ideal. Because the point spread function was not characterized, the exact extent of its influence was not determined.

When considering the position of the motion blur measurements in the wider research context, they appear to be a unique resource. The confirmation of the behaviour of motion blur

is not surprising — other methods relying on the rectangular filter model and the integrating camera (eg [42]) provide implicit verification. However, no other work has been found in the research explicitly recording the relative importance of accounting for different effects which influence image formation. This kind of information is usually considered "common knowledge." The results in Chapter 3 provide evidence of the relative importance of motion blur, lens distortion, noise, and anonymous in-camera processing for a selection of cameras. Also, the sensitivity to noise and spectral occupancy noted in these experiments means this experimental design can be used as a measure of the noise and sharpness performance of a camera. Determining these properties otherwise requires specialised equipment, whereas the experimental design proposed uses only everyday equipment.

Having gathered evidence for the formation of motion blur, the Thesis then considered whether knowledge of the behaviour of motion blur could be used to improve matching results. In Chapters 5 and 6 results were presented which tested the principle that determining the effect of motion blur on a matching process and correcting for it could yield significant improvements in performance. Velocity corrected phase correlation (VCPC) has been shown to improve template-matching precision compared to ordinary phase correlation. Speculative vector-space flattening (SVSF) has been shown to improve descriptor-matching precision compared to ordinary SIFT. The largest improvement (60%) due to SVSF has been shown to be provided in the critical cases where ordinary SIFT produced the fewest matches.

The work described in Chapter 5 provides experimental validation of the method due to Ojansivu and Heikkilä [19] on the only known test material for examining visual tracking performance in the presence of motion blur. Both VCPC and Ojansivu and Heikkilä's method are thereby put into the current context of visual tracking research.

The experiments on SVSF and other modification to SIFT yield significant improvements in the matching of image regions containing motion blur with those which contain no blur, using SIFT features. Gauglitz et al [1] showed that SIFT was amongst the best at this task in their comparison of commonly used image feature descriptors, and the results on their data presented here show a significant improvement.

7.1 Future Work

The general principle of determining the impact of motion blur on some matching process and mitigating it has been shown to be effective on two different methods for image region matching. There is further work to be done in the application of this principle to other contexts. The literature review identified the descriptor in [50], based on the Dual-Tree Complex Wavelet transform as one possible candidate. Gauglitz et al's review [1] found that SURF

descriptors performed best overall when matching a unblurred image features with features containing motion blur. SURF was not investigated as part of this work because the simpler approaches enabled by the structure of Phase Correlation and SIFT meant for more straightforward analysis. Now the principle has been shown to work in two cases, attempting to find an application to SURF [30] descriptors and Phase based local features [52] should follow.

An improved data set containing differences in motion blur could be produced. This may improve the precision available from the experimets described in Chapters 5 and 6. The Sintel open source animated movie [100] could be used to produce frames with realistic motion blur, with free control over the camera parameters leading to blur. The frames would show animation, rather than real scenes, but the differences in image quality due to noise, spatial occupancy, and so on could be analysed and described. Such differences would likely be worthwhile, compared to the control available from the virtual camera.

Gauglitz et al [1] found that image feature detectors are not very reliable under changes in motion blur. It may be possible to apply the principle to the design of some detectors, too.

VCPC might be extendable to a method to estimate the motion blur orientation, which is not recoverable from [19]. In addition, the effect of anti-aliasing, or other more sophisticated methods of generating the mask for VCPC would be interesting, and likely to improve results even further.

The experiments in Chapter 6 showed significant gains in precision. But the perfect detector model is quite unrealistic. Based on these results it is not possible to say that SVSF SIFT matching would provide any significant benefit to a broadcast or feature film virtual graphics system. More work is needed, using a real feature detector, to say with any certainty.

Bibliography

[1] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *International Journal of Computer Vision*, vol. 94, no. 3, pp. 335–360, Mar. 2011.

[2] N. Anantrasirichai, J. Burn, and D. R. Bull, "Robust texture features for blurred images using Undecimated Dual-Tree Complex Wavelets," in *IEEE International Conference on Image Processing*, 2014, pp. 5696–5700.

[3] S. Su, M. Delbracio, J. Wang, G. Sapiro, W. Heidrich, and O. Wang, "Deep video deblurring for hand-held cameras," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1279–1288.

[4] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, Oct. 2005.

[5] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2007.

[6] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[7] A. Barber, D. Cosker, O. James, T. Waine, and R. Patel, "Camera tracking in visual effects: an industry perspective of structure from motion," in *Proceedings of the Symposium on Digital Production*. New York, New York, USA: ACM Press, 2016, pp. 45–54.

[8] http://www.redbeemedia.com/piero.

[9] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[10] J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE Comput. Soc. Press, 1994, pp. 593–600.

[11] G. A. Thomas, "Real-time camera tracking from patches of rich texture," BBC R&D, Tech. Rep. 3379, Dec. 2011.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[13] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, vol. 15, 1988, p. 50.

[14] R. Dawes, J. Chandaria, and G. Thomas, "Image-based camera tracking for athletics," in *Broadband Multimedia Systems and Broadcasting. IEEE International Symposium on*, 2009, pp. 1–6.

[15] http://www.fascinate-project.eu.

[16] O. Schreer, I. Feldmann, C. Weissig, A. Finn, J. Steurer, G. Thomas, A. Gibb, J. Spille, A. Kochale, H. Kropp, M. Borsum, J. Ruiz-Hidalgo, R. Oldfield, B. Shirley, J.-F. Macq, N. Verzijp, M. Prins, S. Matthew, O. Niamut, G. Kienast, R. Kaiser, W. Weiss, and W. Bailer, "Report on final demonstration. fascinate deliverable d6.3.1," 2013.

[17] O. Schreer, P. Kauff, J.-F. Macq, P. Rondão Alface, J. Spille, R. Oldfield, and G. Thomas, "Final specification of generaic data representation and coding scheme," 2013, available from http://www.fascinate-project.eu.

[18] W. Bailer, M. Thaler, F. Lee, R. Oldfield, G. Thomas, and H. Fraser, "Automated metadata extraction tools," 2013, availability restricted.

[19] V. Ojansivu and J. Heikkilä, "Blur insensitive texture classification using local phase quantization," in *International conference on image and signal processing.* Springer, 2008, pp. 236–243.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[21] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," in *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*, September 1980, no. CMU-RI-TR-80-03.

[22] Z. Chen and S.-K. Sun, "A Zernike moment phase-based descriptor for local image representation and matching," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 205–219, 2010.

[23] N. Kingsbury, "Rotation-invariant local feature matching with complex wavelets," in *European Conference on Signal Processing*, 2006, pp. 901–904.

[24] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - A modern synthesis," vol. 1883, no. Chapter 21, pp. 298–372, Apr. 2002.

[25] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *Journal of applied statistics*, vol. 21, no. 1-2, pp. 225–270, 1994.

[26] Y. Dufournaud, C. Schmid, and R. Horaud, "Matching images with different resolutions," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2000, pp. 612–618.

[27] F. Schaffalitzky, T. Kadir, K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, and L. Van Gool, "A comparison of affine region detectors," *International Journal of Computer Vision*, vol. 65, no. 1-2, pp. 43–72, Oct. 2005.

[28] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *European Conference on Computer Vision*, May 2002, pp. 128–142.

[29] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE International Conference on Computer Vision*. IEEE, 1999, pp. 1150–1157.

[30] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[31] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions," *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, 2004.

[32] T. Tuytelaars and L. Van Gool, "Matching widely separated views based on affine invariant regions," *International Journal of Computer Vision*, vol. 59, no. 1, pp. 61–85, Apr. 2004.

[33] ——, "Content-based image retrieval based on local affinely invariant regions," in *Visual Information and Information Systems*, 1999, pp. 493–500.

[34] T. Kadir and M. Brady, "Scale saliency: A novel approach to salient feature and scale selection," in *IET International Conference on Visual Information Engineering*, 2003, pp. 25–28.

[35] C. D. Kuglin and D. C. Hines, "The phase correlation image alignment method," in *Proceeding of IEEE International Conference on Cybernetics and Society*, 1975, pp. 163–165.

[36] V. Ojansivu and J. Heikkilä, "Image registration using blur-invariant phase correlation," *Signal Processing Letters, IEEE*, vol. 14, no. 7, pp. 449–452, 2007.

[37] C. Tomasi and T. Kanade, "Detection and tracking of point features," Pittsburgh, 1991.

[38] S. Baker and I. Matthews, "Lucas-Kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.

[39] http://www.bbc.co.uk/rd/projects/piero.

[40] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 943–948.

[41] T. Drummond and R. Cipolla, "Visual tracking and control using Lie algebras," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, 1999, pp. 652–657.

[42] Y. Park, V. Lepetit, and W. Woo, "ESM-Blur: Handling & rendering blur in 3D tracking and augmentation," in *IEEE International Symposium on Mixed and Augmented Reality.* IEEE Computer Society, 2009, pp. 163–166.

[43] ——, "Handling motion-blur in 3D tracking and rendering for augmented reality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 9, pp. 1449–1459, 2012.

[44] H. Jin, P. Favaro, and R. Cipolla, "Visual tracking in the presence of motion blur," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 18–25.

[45] C. Mei and I. Reid, "Modeling and generating complex motion blur for real-time tracking," in *IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, Jun. 2008, pp. 1–8.

[46] S. Dai, M. Yang, Y. Wu, and A. K. Katsaggelos, "Tracking motion-blurred targets in video," in *IEEE International Conference on Image Processing*, 2006, pp. 2389–2392.

[47] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, 2004, pp. 506–513.

[48] P. Mainali, G. Lafruit, Q. Yang, B. Geelen, R. Lauwereins, and L. Van Gool, "SIFER: Scale-invariant feature detector with error resilience," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 172–197, Apr. 2013.

[49] Y. Yu, K. Huang, W. Chen, and T. Tan, "A novel algorithm for view and illumination invariant image matching," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 229–240, 2012.

[50] P. Bendale, B. Triggs, and N. Kingsbury, "Multiscale keypoint analysis based on complex wavelets," in *British Machine Vision Conference*. British Machine Vision Association, 2010, pp. 49.1–49.10.

[51] N. Kingsbury, "Complex wavelets for shift invariant analysis and filtering of signals," *Applied and Computational Harmonic Analysis*, vol. 10, no. 3, pp. 234–253, 2001.

[52] G. Carneiro and A. D. Jepson, "Phase-based local features," in *European Conference on Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 282–291.

[53] E. Rahtu, J. Heikkilä, V. Ojansivu, and T. Ahonen, "Local phase quantization for blur-insensitive image analysis," *Image and Vision Computing*, vol. 30, no. 8, Aug. 2012.

[54] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello, "A visual odometry framework robust to motion blur," in *IEEE International Conference on Robotics and Automation*. IEEE, Mar. 2009, pp. 2250–2257.

[55] W. H. Richardson, "Bayesian-based iterative method of image restoration," *Journal of the Optical Society of America*, vol. 62, no. 1, pp. 55–59, 1972.

[56] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *The astronomical journal*, vol. 79, p. 745, 1974.

[57] Y. Zhang and K. Hirakawa, "Blind deblurring and denoising of images corrupted by unidirectional object motion blur and sensor noise," *IEEE transactions on image processing*, vol. 25, no. 9, pp. 4129–4144, 2016.

[58] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *IEEE international conference on computer vision*. IEEE, 2011, pp. 479–486.

[59] M. J. Shah and U. D. Dalal, "Blind estimation of motion blur kernel parameters using Cepstral domain and Hough transform," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, 2014, pp. 992–997.

[60] D. A. Fish, A. M. Brinicombe, and E. R. Pike, "Blind deconvolution by means of the Richardson-Lucy algorithm," *Journal of the Optical Society of America*, vol. 12, no. 1, pp. 58–65, 1995.

[61] D. Perrone and P. Favaro, "A clearer picture of total variation blind deconvolution," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 6, pp. 1041–1055, 2016.

[62] T. F. Chan and C.-K. Wong, "Total variation blind deconvolution," *IEEE transactions on Image Processing*, vol. 7, no. 3, pp. 370–375, 1998.

[63] H. Takeda and P. Milanfar, "Removing motion blur with space–time processing," *IEEE transactions on image processing*, vol. 20, no. 10, pp. 2990–3000, 2011.

[64] M. Delbracio and G. Sapiro, "Burst deblurring: Removing camera shake through Fourier burst accumulation," in *IEEE Conference on computer vision and pattern recognition*. IEEE, 2015, pp. 2385–2393.

[65] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 2414–2423.

[66] A. Dosovitskiy, J. T. Springenberg, and T. Brox, "Learning to generate chairs with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015, pp. 1538–1546.

[67] Inceptionism: Going Deeper into Neural Networks. [Online]. Available: https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[68] M. Noroozi, P. Chandramouli, and P. Favaro, "Motion deblurring in the wild," in *German Conference on Pattern Recognition*. Springer, 2017, pp. 65–77.

[69] A. Chakrabarti, "A neural approach to blind motion deblurring," in *European conference on computer vision*. Springer, 2016, pp. 221–235.

[70] T. Portz, L. Zhang, and H. Jiang, "Optical flow in the presence of spatially-varying motion blur," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 1752–1759.

[71] T. H. Kim, S. Nah, and K. M. Lee, "Dynamic scene deblurring using a locally adaptive linear blur model," *arXiv preprint arXiv:1603.04265*, 2016.

[72] M. H. Daraei, "Optical flow computation in the presence of spatially-varying motion blur," in *International Symposium on Visual Computing*. Springer, 2014, pp. 140–150.

[73] Y. Schoueri, M. Scaccia, and I. Rekleitis, "Optical flow from motion blurred color images," in *Computer and Robot Vision, 2009. CRV'09. Canadian Conference on*. IEEE, 2009, pp. 1–7.

[74] Z. Tu, R. Poppe, and R. Veltkamp, "Estimating accurate optical flow in the presence of motion blur," *Journal of Electronic Imaging*, vol. 24, no. 5, p. 053018, 2015.

[75] W. Li, Y. Chen, J. Lee, G. Ren, and D. Cosker, "Robust optical flow estimation for continuous blurred scenes using rgb-motion imaging and directional filtering," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*. IEEE, 2014, pp. 792–799.

[76] M. Okade and P. K. Biswas, "Improving video stabilization in the presence of motion blur," in *Third National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*. IEEE, 2011, pp. 78–81.

[77] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single photograph," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 787–794, 2006.

[78] Y. Wu, H. Ling, J. Yu, F. Li, X. Mei, and E. Cheng, "Blurred target tracking by Blur-driven Tracker," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 1100–1107.

[79] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[80] T. Kadir, A. Zisserman, and M. Brady, "An affine invariant salient region detector," in *European Conference on Computer Vision*. Springer, 2004, pp. 228–241.

[81] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*. Springer, 2006, pp. 404–417.

[82] "ITU-R BT.709: Parameter values for the HDTV standards for production and international programme exchange," Jul. 2009.

[83] A. Roberts, *Circles of Confusion*, 1st ed. European Broadcasting Union, Aug. 2009.

[84] J. J. Koenderink and A. J. Van Doorn, "Affine structure from motion," *Journal of the Optical Society of America*, vol. 8, no. 2, pp. 377–385, 1991.

[85] K. Schelten and S. Roth, "Localized image blur removal through non-parametric kernel estimation," in *2014 22nd International Conference on Pattern Recognition (ICPR)*. IEEE, pp. 702–707.

[86] G. Bradski, "The opencv software library," *Dr. Dobb's Journal of Software Tools*, 2000.

[87] G. A. Thomas, "Motion estimation and its applications in broadcast television," Ph.D. dissertation, University of Essex, 2 1990.

[88] S. C. Dabner, "Real-time motion measurement hardware: Phase correlation unit," BBC Research Department, Tech. Rep. 1990/11.

[89] "The fastest fourier transform in the west," http://fftw.org.

[90] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2911–2918.

[91] G. S. Klein and T. Drummond, "A Single-frame visual gyroscope," in *British Machine Vision Conference*. British Machine Vision Association, 2005.

[92] M. Ben-Ezra and S. K. Nayar, "Motion-based motion deblurring," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, Jun. 2004.

[93] D. L. Donoho and M. E. Raimondo, "A fast wavelet algorithm for image deblurring," *ANZIAM Journal*, vol. 46, no. 0, pp. 29–46, Mar. 2005.

[94] A. Levin, "Blind motion deblurring using image statistics," in *Advances in Neural Information Processing Systems 19:Proceedings of the 2006 Conference*, pp. 841–848.

[95] Y.-W. Tai, P. Tan, and M. S. Brown, "Richardson-Lucy deblurring for scenes under a projective motion path," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1603–1618, 2011.

[96] A. Vedaldi and B. Fulkerson, "VLfeat: an open and portable library of computer vision algorithms," in *MM '10: Proceedings of the international conference on Multimedia*. ACM Request Permissions, Oct. 2010.

[97] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *CoRR*, vol. abs/1411.1607, 2014. [Online]. Available: http://arxiv.org/abs/1411.1607

[98] S. Dai and Y. Wu, "Motion from blur," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[99] J. Fauqueur, N. Kingsbury, and R. Anderson, "Multiscale keypoint detection using the dual-tree complex wavelet transform," in *Image Processing, IEEE International Conference on*, 2006, pp. 1625–1628.

[100] "Sintel, the durian open movie project," https://durian.blender.org.